Peer-to-Peer Netw. Appl. (2013) 6:61–74
DOI 10.1007/s12083-012-0129-7

Practical authentication scheme for SIP

Shuhua Wu · Qiong Pu · Fei Kang

Received: 9 September 2011 / Accepted: 1 March 2012 / Published online: 15 April 2012

Abstract The Session Initiation Protocol (SIP) is commonly used to establish Voice over IP (VoIP) calls. However, the original authentication scheme for SIP-based service typically uses HTTP Digest authentication protocol, which is s not providing security at an acceptable level. In this paper, we propose a secure and practical password-only authenticated key agreement scheme for SIP using elliptic curve cryptography(ECC). Our scheme is remarkable efficient and quite simple to use. And yet we can provide the rigorous proof of the security for it. Therefore, the end result is more suited to be a candidate for SIP authentication scheme. In addition, we also suggest an extended scheme capable of providing anonymity, privacy, and location privacy to protect the user's personal information and his real identity.

Keywords VoIP · SIP · Elliptic curve · Authentication · Password

1 Introduction

Voice over Internet Protocol (VoIP) is a fast growing technology believed to be the future replacement for traditional Public Switched Telephone Network (PSTN) networks. There are many protocols used in VoIP signaling, but Session Initiation Protocol (SIP) is one of the widely used ones. It has been chosen by the Third-Generation Partnership Project (3GPP) as the protocol for multimedia application in 3G mobile networks.

The Session Initiation Protocol [1, 2] is an application-layer protocol that is capable of handling all the signalling requirements of a VoIP session, i.e. initiating, managing and terminating voice and video sessions across packet networks. It is analogous to the SS7 [3] protocol in traditional telephony. Security and privacy requirements in a VoIP environment are expected to be equivalent to those in PSTN. However, the original authentication scheme for SIP-based service typically uses HTTP Digest authentication protocol [4], which is not providing security at an acceptable level [5–8]. Many of research efforts have been directed to investigate the authentication protocol for SIP, with a keen focus on security enhancements. A great deal of improvements have been studied in the existing literature [4, 9–18]. However, all of these schemes have one or more of the following disadvantages, which may hamper the wide application in practice.

– The authentication mechanism in [10, 12, 13, 15] depends on a cryptographic key. As a result, an extra smart card should be used to store it since human users cannot remember or securely store long, high-entropy keys. As noted in [19], in the real world, despite the recognition of their functionalities and security, smart cards have not yet prevailed. The high cost of the cards and readers remains a burden to issuers or users. In addition, there are more subtle

S. Wu (✉) · F. Kang
Department of Networks Engineering, Information Engineering University, Zhengzhou, China
e-mail: tybhsl@yahoo.com.cn

S. Wu
State Key Laboratory of Information Security, Graduate University of Chinese Academy of Sciences, Beijing, China

Q. Pu
CIMS Research Center, Tongji Univerity, Shanghai, China

problems in deploying the necessary infrastructure for smart cards, including the method of uploading different secure access modules into card readers. These obstacles have restricted the application of smart cards to the small fields.

- The schemes in [12, 13, 16] involve the third trust party (TTP). All the PKI-base authentication schemes, such as SIPS (SIP over SSL), need a trust authority providing certificates for end-user applications. All the authentication scheme using identity-based cryptography [20], or certificateless public key cryptography [21], or using self-certified public key cryptography [22, 23], such as [12, 13, 16] respectively, require a trust authority involved in key generation. These schemes are seriously limited in that there is virtually no consolidated authority today that can play a role of that kind. Therefore, they can only be applied in a small scale or closed domains. Furthermore, these schemes usually suffers from the heavy computation load. This is true especially for the pairing-based schemes [12, 13].
- The password-based authentication schemes proposed in [4, 9, 11, 14] are insecure although they do avoid the two aforesaid defects. As noted in [16], the schemes in [4, 14] are insecure since the offline password guessing attack can not be avoided . As for Yang et al's scheme in [11], it is either insecure because it incurs the replay attack. Moreover it is so inefficient because it is proposed only for Discrete Logarithm (DL) settings and involves in costly exponential computation, which is not suitable for the user's device with limited computing capability. Their scheme makes no use of the good characteristics of Elliptic Curve(EC) at all. Please note, like the sheme [9], the direct EC analog of the DL based protocol proposed by Yang et al. [11] is completely insecure as it can't resist the offline password guessing attack based on the results in [24].

In this paper, we propose a secure and practical password authenticated key agreement scheme for SIP using elliptic curve cryptography [25, 26]. Our scheme is remarkable efficient and is suitable for the user's device with limited computing capability. And it is quite simple to use. Our scheme does not need the third trust party any more. Instead, we make use of SIP architecture itself to achieve password-only authentication. In addition, we can provide the rigorous proof of the security for our scheme. Finally, to protect the user's personal information and his real identity, we also suggest an extended scheme capable of providing anonymity, privacy, and location privacy. Therefore, the end result is more suited to be a candidate for SIP authentication scheme.

The remainder of this paper is organized as follows. Section 2 reviews SIP authentication scheme and its security issues briefly. Section 3 provides an improved scheme to overcome all those disadvantages existing in previous schemes. In addition, some important discussions are also made in this section. Section 4 provides the rigorous proof of the security for our scheme. Finally, conclusion is presented in Section 5.

2 SIP authentication

In this section, we briefly review SIP protocol and SIP authentication scheme. Thereafter, we revisit the security issues related to SIP authentication.

2.1 SIP protocol

We briefly introduce the SIP architecture [27] and then give an example [8] to illustrate the call procedure. And we follow the description in [8, 27].

SIP is an application-layer signaling protocol [1, 2] for handling multimedia sessions over the Internet. A VoIP media session is, by nature, a peer-to-peer connection, i.e. a terminal device/software should be able to contact another terminal without intermediaries involved. However, the called party needs to be located on the Internet before a media session can be established. Therefore, in addition to terminal device/software, the following network elements are also involved: proxy/ redirect servers, registrar and location service, which are the logical entities responsible for the discovery process in a certain domain. Figure 1 shows how different logical entities interact with each other in SIP protocols. A SIP phone (terminal device or softphone) registers its contact address (address of the host where it is located) to registrar servers, which then stores the address binding with location services. During a call setup, the calling party sends its invitation request to his local outbound proxy. The outbound proxy finds out the proxy that is responsible for the destination domain through a particular type of DNS lookup [28], and forwards the request to it. The destination domain proxy will query its location service for the called party's current contact address and then forward the request to that address. The callee's response message goes through the same channel in the reverse direction back to the caller. After finalizing the signaling session establishment, the end systems can

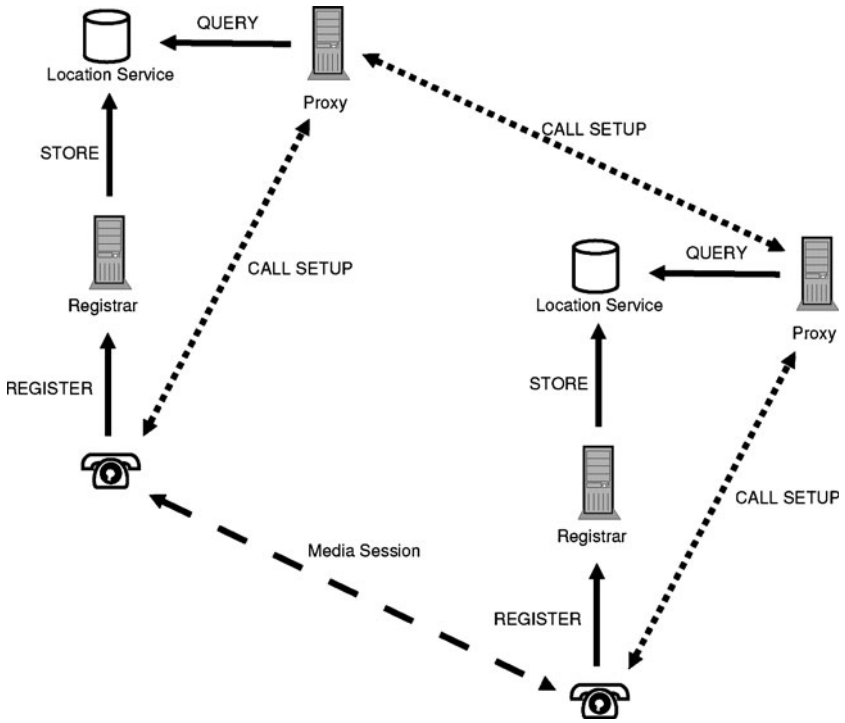

Fig. 1 SIP interactions[27]

exchange media data directly without the involvement of any SIP proxy.

Redirect server is just used to inform the caller to access backup proxy when the corresponding proxy is temporarily unavailable. And redirect server is not shown in the figure.

As an example call flow, consider user Alice and user Bob as introduced in [8]. SIP user identification is based on a special type of Uniform Resource Identifier (URI) called a SIP URI with a form similar to an email address. In the following example Alice's SIP URI is assumed to be sip:alice@atlanta.com and Bob's sip:bob@biloxi.com. Figure 2 [8] shows a typical

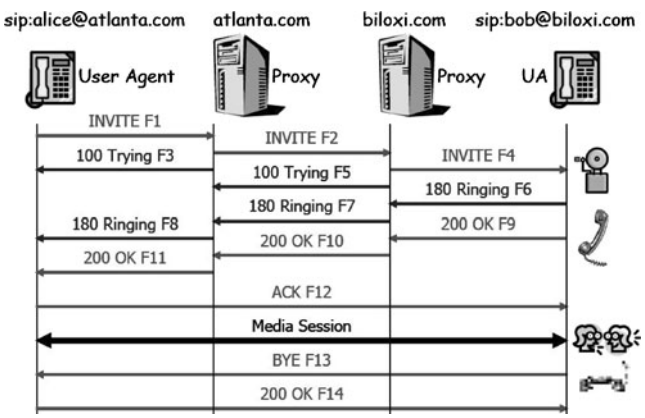

Fig. 2 Session initiation between two user agents [8]

SIP message exchange scenario between two users Alice and Bob belonging to the domains atlanta.com and biloxi.com, respectively. In the example, the atlanta.com proxy will make a DNS lookup to determine the proxy server of the biloxi.com domain on behalf of Alices user agent. The SIP INVITE request originating from Alices UA is then forwarded via the atlanta.com proxy to the biloxi.com proxy which with the help of a location service determines the current whereabouts of Bobs user agent. Both the informational Ringing message and the OK message which is issued when Bob accepts the call, take the return path via the proxy server hops whereas the ACK message and the payload packets of the ensuing multimedia session will use the direct path between the two user agents. The ongoing session ends, when either one sends an BYE message to the other peer. Figure 2 includes all necessary information required to set up an audio connection.

2.2 SIP authentication

Using SIP, there exists two basic principles for providing security, end-to-end and hop-by-hop [10]. End-to-end security on SIP data involves end users, for example SIP authentication. In contrast to end-to-end solutions which use SIP mechanisms to ensure security, hop-by-hop relies on the security provided by the network. Examples of hop-by-hop mechanism are

transport-level security (TLS) and Internet Protocol Security (IPsec) [5]. This paper is only interested in the SIP authentication mechanisms.

In SIP specification [1], the authentication mechanism proposed is HTTP digest based authentication. In SIP terms, HTTP digest mechanism is called the SIP authentication. SIP applies the digest mechanism for authenticating users to users or users to proxies/registars, not proxies to proxies(The security between proxies relies on other mechanisms, for example TLS or IPsec.). Next we follow the description [5] to introduce it. HTTP Digest authentication [4] is a challenge-based mechanism: when a server receives a request, it may challenge the initiator of the request to provide assurance of its identity. The challenge contains a nonce value that is a string uniquely generated and used for one challenge only. Both the requester and the server share a secret password, and the requester uses this secret password, together with the nonce value, to compute a response value. The requester sends the request again, including the computed response value, which is used by the server to authenticate the request. A representation of the digest authentication procedure is given in Fig. 3, where the function F used to compute the response specifies how to combine the input parameters with some iterations of a digest algorithm. The adaptation of this procedure to SIP is straightforward except that one more flow is added to inform the client that the authentication succeeds. The authentication procedure is run when the callee, an intermediate proxy server, or the registrar server requires the caller to be authenticated before accepting the call, forwarding the call, or accepting the registration.

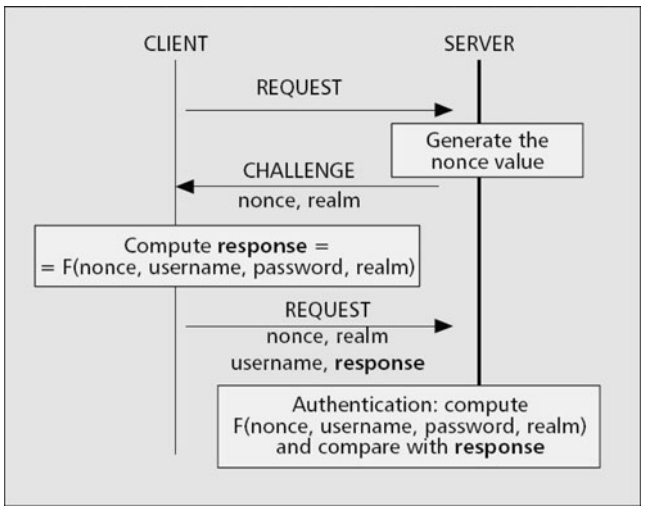

Fig. 3 Digest authentication [5]

2.3 SIP security issues

This authentication scheme can offer one-way message authentication and replay protection. However, it has the following disadvantages [5–8].

- Firstly, the mutual authentication is not provided since there is no any procedure such that the client can authenticate the server. Therefore, it is possible for a malicious attacker to place sever spoofing attack.
- Secondly, it cannot support message integrity and confidentiality to secure all headers and parameters in SIP which would possibly need protection. There are no session keys established in the digest authentication; therefore, it does not provide confidentiality protection on SIP messages. In addition, the integrity mechanisms in Digest do not work very well for SIP since it offers protection only for some SIP parameters, leaving unauthenticated several header fields user agents might wish to secure.
- Thirdly, this method is vulnerable to off-line password guessing attack since the response value consists of the digest of the user's password combined with some parameters that can be easily captured by a potential aggressor simply by sniffing the network traffic.
- Lastly but not least, it can not provide end-to-end protection, and it is difficult to expand beyond a single administration domain since it is based on shared user passwords. If no end-to-end protection mechanism is in place, many kinds of attacks, such as modification, eavesdropping, session disruption, imitation, and so forth, can also be launched successfully during the transmission of the media packets (e.g., voice). Moreover, an attacker can possibly utilize the BYE request to tear down a session in this case. This kind of attack is known as signaling attack.

3 Our proposed authentication scheme

For simplicity, and without loss of generality, we assume an instance of SIP proxy co-locates with an instance of SIP registrar and SIP location service on a single host. This is so because all these entities in the general case are connected together through a secure communication channel. Thus we consider here a simple topology as described in Fig. 4 with a SIP server responsible for a certain domain.

From the point of view of network security this means that every channel in the figure(i.e., Channel 0,

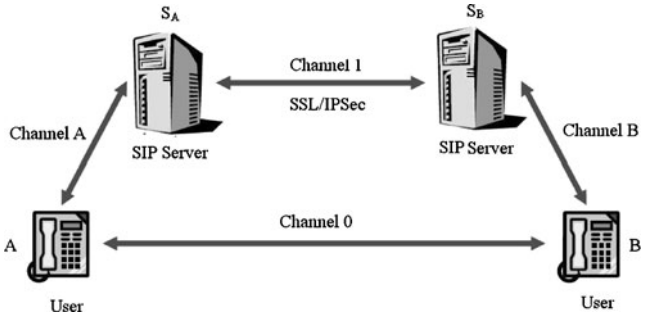

Fig. 4 Simplified topology for SIP

Channel 1, Channel A, Channel B) must be secured. Since the security of Channel 1 relies on other mechanisms, for example TLS or IPsec, which is uninterested in this paper, we assume it is already an authenticated and secure channel while the other communication channels are no longer considered to be secure in this paper. We further assume all servers are non-malicious; and SIP servers in the caller domain and the callee domain trust each other to correctly establish contact identity and address binding for their own domains.

Before we introduce our scheme indeed, we first define some notations used in our scheme in Table 1.

Please note, to be applicable in low resource environments, we implement it over elliptic curve (EC) because of the well-known advantages with regard to processing and size constraints [25, 26].

3.1 Description

Now we come to introduce our scheme indeed. As a general scenario, our scheme mainly consists of two phases, i.e. registration and call setup, which are used by a user to identify its location and establish a call respectively. Both of them follow the way of handshake in SIP protocol. In subsequent sections, we will present them in detail. Please note the representation is slightly simplified with respect to the message parameters. Especially, all the all headers and parameters in SIP that need protection in integrity should be included in the $\mathcal{H}$'s input at the computation of the message authenticator.

Table 1 The notations used in our scheme

$\mathcal{E}$	An elliptic curve defined over a prime finite field $\mathbb{F}_p$ with large order
P	A base point in $\mathcal{E}$ with large order q, where q is a secure large prime
$\mathbb{G}$	A cyclic additive group generated by P
$x \cdot P$	The point multiplication defined as $x \cdot P = \underbrace{P + P + \cdots + P}_{x \text{ times}}$
$E_k(\cdot)/D_k(\cdot)$	A symmetric encryption/decryption algorithm, where k denotes the symmetric key;
$\mathcal{G}(\cdot)$	A secure one-way hash function: $\{0,1\}^* \rightarrow \mathbb{G}$
$\mathcal{H}(\cdot)$	A secure one-way hash function: $\{0,1\}^* \rightarrow \{0,1\}^l$, where l is the secure parameter.

3.1.1 Registration procedure

A user A registers the combination of his/her name(or SIP identity) A and current IP address AR_A at the SIP Server S_A responsible for his domain. This registration procedure follows, as is depicted in Fig. 5, where $PW_A = \mathcal{G}(pw_A)$. Our protocol is based on the password-based authenticated key exchange in [30].

Step 1. A sends to S_A a REGISTER, which includes both AR_A and A.

Step 2. On receiving the REGISTER, S_A chooses a random number $x \in Z_q^*$, computes its nonce $X^\star = xP + PW_A$, and then sends to A a CHALLENGE including its name(or realm) S_A and its nonce $X^\star$. Please note, in SIP, CHALLENGE is actually sent in form of an error message "Unauthorised".

Step 3. On receiving the CHALLENGE, A chooses a random number $y \in Z_q^*$, computes its nonce $Y = yP$, and then sends to S_A a RESPONSE including its initial REGISTER along with its nonce Y and its authenticator $\text{Auth1}_{AS} = \mathcal{H}(\text{"1"} \| A \| S_A \| Y \| X^\star \| PW_A \| AR_A \| Z_{AS})$, where $Z_{AS} = y(X^\star - PW_A)$.

Step 4. After the RESPONSE is received, S_A computes $Z_{AS} = xY$ and then verifies $\mathcal{H}(\text{"1"} \| A \| S_A \| Y \| X^\star \| PW_A \| AR_A \| Z_{AS}) \stackrel{?}{=} \text{Auth1}_{AS}$. If they are unequal, S_A terminates the procedure. Otherwise, S_A accepts A's registration, and then sends to A an OK including its authenticator $\text{Auth1}_{SA} = \mathcal{H}(\text{"2"} \| S_A \| A \| X^\star \| Y \| PW_A \| AR_A \| Z_{AS})$. Finally, S_A computes the session key $K_{AS} = \mathcal{H}(\text{"0"} \| S_A \| A \| X^\star \| Y \| PW_A \| AR_A \| Z_{AS})$.

Step 5. On receiving OK, A verifies $\mathcal{H}(\text{"2"} \| S_A \| A \| X^\star \| Y \| PW_A \| AR_A \| Z_{AS}) \stackrel{?}{=} \text{Auth1}_{SA}$. If they are equal, A believes his registration to S_A is successful and then proceeds to compute the session key $K_{AS} = \mathcal{H}(\text{"0"} \| S_A \| A \| X^\star \| Y \| PW_A \| AR_A \| Z_{AS})$.

When a successful registration occurs, the server S_A will establish a pair of counters (N_{SA}, XN_{AS}) for the user A, and the user will also yield a pair of counters (N_{AS}, XN_{SA}) associated with S_A. The initial values

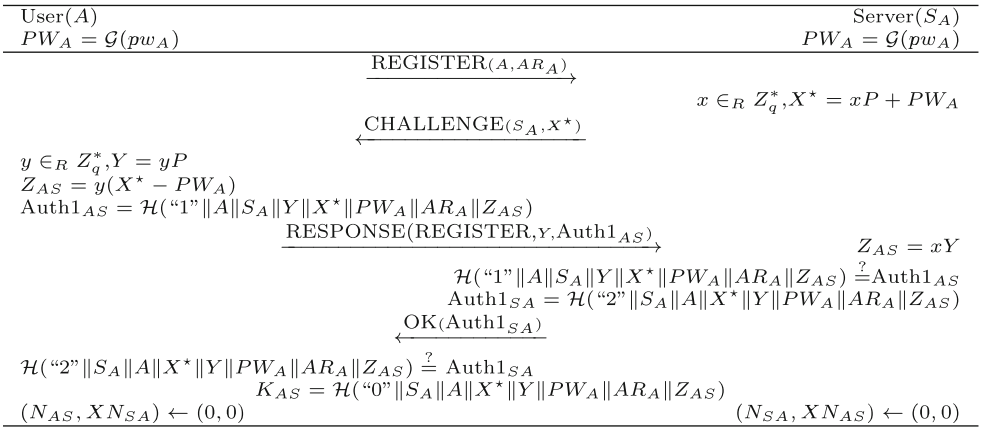

Fig. 5 Registration phrase of our scheme

for them are set to zeroes. These counters are used to prevent replay-attack in call setup(to be explained later). When the registration expires in certain time, for example in a matter of hours, and these counters as well as the session key K_{AS} will also expire. In that case, a new registration is needed.

3.1.2 Call setup

We assume the two users A and B have successfully registered to S_A and S_B respectively. Then A will share a session key K_{AS} with S_A . And A will yield a pair of counters (N_{AS}, XN_{SA}) and S_A will have (N_{SA}, XN_{AS}), where N_{AS} (resp. N_{SA}) records the number of SIP messages sent from A (resp. S_A) to S_A (resp. A) and XN_{SA}(resp. XN_{AS}) represents the maximum value of N_{AS}(resp. N_{SA}) received by A (resp. S_A). Similarly, B will share a session key K_{BS} with S_B. And B will yield a pair of counters (N_{BS}, XN_{SB}) and S_B will have (N_{SB}, XN_{BS}). When A wants to invite B to participate to a call, then the following procedure is deployed, as is depicted in Fig. 6.

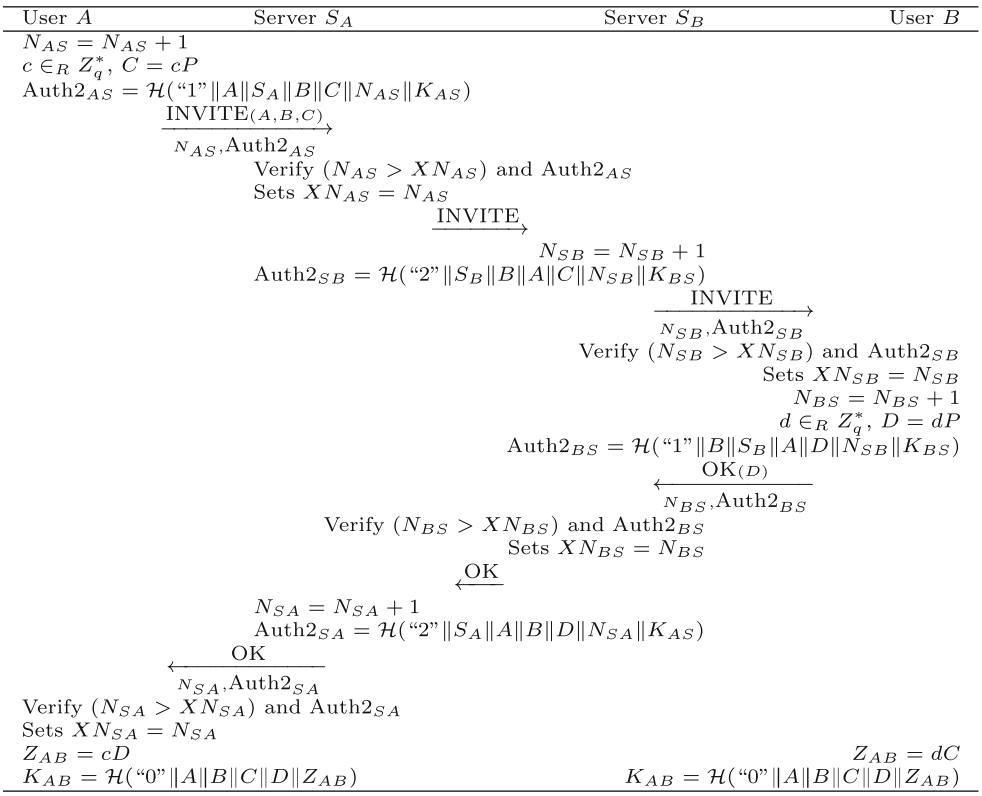

Fig. 6 Call setup of our scheme

1. A first increases N_{AS} by 1 to update its value. And A chooses a random number $c \in Z_q^*$, computes its nonce $C = cP$, and then sends to S_A an INVITE(which includes the caller's name A, the callee's name B and its nonce C) along with the value N_{AS} and A's authenticator $\text{Auth2}_{AS} = \mathcal{H}(\text{“1”}\|A\|S_A\|B\|C\|N_{AS}\|K_{AS})$.
2. On receiving the INVITE as well as N_{AS} and Auth2_{AS}, S_A first checks if both $N_{AS} > XN_{AS}$ and $\mathcal{H}(\text{“1”}\|A\|S_A\|B\|C\|N_{AS}\|K_{AS}) \stackrel{?}{=} \text{Auth2}_{AS}$ hold. If either of them fails, S_A will halt the protocol run. Otherwise, S_A sets $XN_{AS} = N_{AS}$ and forwards the INVITE to S_B.
3. On receiving the INVITE, S_B first increases N_{SB} by 1 to update its value and computes its authenticator $\text{Auth2}_{SB} = \mathcal{H}(\text{“2”}\|S_B\|B\|A\|C\|N_{SB}\|K_{BS})$. Then S_B retrieves B's address in the database and forwards to B the INVITE along with the value N_{SB} and its authenticator Auth2_{SB}.
4. On receiving the INVITE as well as N_{SB} and Auth2_{SB}, B first checks if both $N_{SB} > XN_{SB}$ and $\mathcal{H}(\text{“2”}\|S_B\|B\|A\|C\|N_{SB}\|K_{BS}) \stackrel{?}{=} \text{Auth2}_{SB}$ hold. If either of them fails, B will halt the protocol run. Otherwise, B sets $XN_{SB} = N_{SB}$, increases N_{BS} by 1 to update its value. Then B chooses a random number $d \in Z_q^*$, computes its nonce $D = dP$ and computes its authenticator $\text{Auth2}_{BS} = \mathcal{H}(\text{“1”}\|B\|S_B\|A\|D\|N_{SB}\|K_{BS})$. Finally, B sends to S_B an OK(which includes its nonce D) along with the value N_{BS} and B's authenticator Auth2_{BS}. In addition he also computes the session key $K_{AB} = \mathcal{H}(\text{“0”}\|A\|B\|C\|D\|Z_{AB})$, where $Z_{AB} = dC$.
5. On receiving the OK as well as N_{BS} and Auth2_{BS}, S_B first checks if both $N_{BS} > XN_{BS}$ and $\mathcal{H}(\text{“1”}\|B\|S_B\|A\|D\|N_{SB}\|K_{BS}) \stackrel{?}{=} \text{Auth2}_{BS}$ hold. If either of them fails, S_B will halt the protocol run. Otherwise, S_B sets $XN_{BS} = N_{BS}$, and forwards the OK to S_A.
6. On receiving the OK, S_A first increases N_{SA} by 1 to update its value and computes its authenticator $\text{Auth2}_{SA} = \mathcal{H}(\text{“2”}\|S_A\|A\|B\|D\|N_{SA}\|K_{AS})$. Then S_A retrieves A's address in the database and forwards to A the OK along with the value N_{SA} and its authenticator Auth2_{SA}.
7. On receiving the OK as well as N_{SA} and Auth2_{SA}, A first checks if both $N_{SA} > XN_{SA}$ and $\mathcal{H}(\text{“2”}\|S_A\|A\|B\|D\|N_{SA}\|K_{AS}) \stackrel{?}{=} \text{Auth2}_{SA}$ hold. If either of them fails, A will halt the protocol run. Otherwise, A sets $XN_{SA} = N_{SA}$ computes the session key $K_{AB} = \mathcal{H}(\text{“0”}\|A\|B\|C\|D\|Z_{AB})$, where $Z_{AB} = cD$.

In the end, A and B will share the common session key K_{AB}, which can be used to protect the signaling data (e.g. ACK or BYE) and media data transmitted end-to-end subsequently.

Please note, for simplicity, we omit the description on how to send Trying and Ringing message. Actually, they can also be sent in an authenticated way similarly. For example, S_A can send Ringing to A as follows: S_A first increases N_{SA} by 1 to update its value and computes its authenticator $\text{Auth3}_{SA} = \mathcal{H}(\text{“3”}\|S_A\|A\|N_{SA}\|K_{AS})$. Then S_A retrieves A's address in the database and forwards to A the Ringing along with the value N_{SA} and its authenticator Auth3_{SA}; after the Ringing as well as N_{SA} and Auth3_{SA} is received, A first checks if both $N_{SA} > XN_{SA}$ and $\mathcal{H}(\text{“3”}\|S_A\|A\|N_{SA}\|K_{AS}) \stackrel{?}{=} \text{Auth3}_{SA}$ hold. If either of them fails, A will ignore this message. Otherwise, A sets $XN_{SA} = N_{SA}$ and waits. By this means, the receiver can ensure that the received message is sent by the intended sender and not replayed by any attacker.

3.2 Privacy considerations

In addition to authentication issues, it is of equal importance to protect the user's personal information and his real identity providing anonymity, privacy, and location privacy. Next, we will mainly show how to protect user's real identity and present a scheme with identity protection. Identity protection means that no one can know the real identity of the client except the server.

To protect user's real identity, each user has to use temporary identity instead of real one to register his address and then change the temporary identity value after each registration. The renewed temporary identity will be used to initiate a call indeed. To achieve this goal, we assume A and S_A share a common initial temporary identity of A, SU. A should remember SU in addition to his password pw_A; and S_A maintains the mapping relating between SU and A by storing two variables SU_A and SU'_A in the database, where SU_A represents A's current temporary identity and SU'_A represents A's previous temporary identity. And the initial values of them are set to be SU. This registration procedure follows, as is depicted in Fig. 7, where $PW_A = \mathcal{G}(pw_A)$:

In the Registration phase of the scheme, A sends to S_A a REGISTER, in which SU instead of A is used and then S_A determines the sender's identity using SU by looking up in the database to find the entity such that $SU_A = SU$ or $SU'_A = SU$. After the CHALLENGE is received, A will choose a

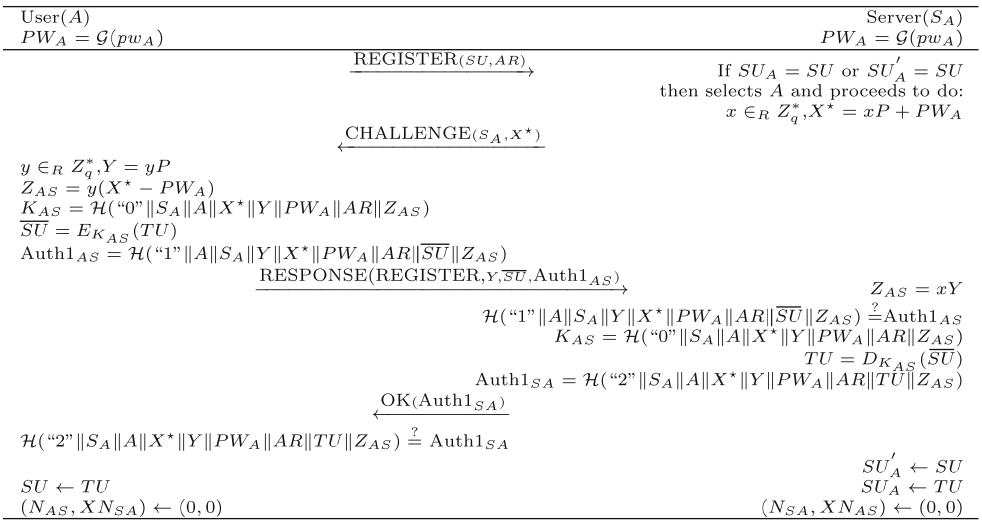

Fig. 7 Registration phrase of our scheme with identity protection

new temporary identity TU, encrypt this value using K_{AS} as $\overline{SU}$ and then send $\overline{SU}$ to S_A in an authenticated way. Later, S_A will recover TU from $\overline{SU}$ and authenticate itself to A. Certainly, the computation of the authenticators should be modified accordingly: i.e. $\text{Auth1}_{AS} = \mathcal{H}(\text{“1”}\|A\|S_A\|Y\|X^\star\|PW_A\|AR\|\overline{SU}\|Z_{AS})$ and $\text{Auth1}_{SA} = \mathcal{H}(\text{“2”}\|S_A\|A\|X^\star\|Y\|PW_A\|AR\|TU\|Z_{AS})$. In the end, S_A will set $SU_A = TU$ and reserve the used temporary identity SU in SU'_A and A will set $SU = TU$. After that, SU with a new value TU will be used to initial a call. To protect the callee's identity, B should be sent in ciphertext $E_{K_{AS}}(B)$ in the request. Then S_A will recognize the identity of A by finding in the database such a record that $SU_A = SU$ holds and use K_{AS} to recover the identity of B. After that, the request with real identities will be forward to S_A and the latter will send to B the request with users' identities encrypted using K_{BS}. When B sends a message to A via S_B then S_A, users' identities should encrypted using K_{BS} and K_{AS} accordingly to hide the real ones. Please note, to prevent an attacker to trace the user's identity, a random string must appended to its end so that an identity can be encrypted as different ciphertext even under the same key.

Like user's real identity, other personal information can also be protected in our scheme. In this case, it should be encrypted using K_{AS} and K_{BS} accordingly before it can be sent in the network.

3.3 Security

Here we just provide the intuitive understanding the security for our scheme. And the rigorous proof of the security can be found in next section. Please note many previous cryptographic schemes containing only informal arguments for security were subsequently shown to be insecure, e.g. [4, 14]. Therefore, the importance of formal proofs of security should be emphasized to design cryptographic protocols.

At first, the protocol given in Section 3.1.1 for registration can provide mutual authentication and establish a secret session key only known by the two communicating parties. Moreover, it can protect the password information against the notorious password guessing attacks by which attackers could search the relatively small space of human-memorable passwords. In other words, it is a secure password-based key agreement protocol with mutual authentication. Next we come to explain it. If the adversary tries to impersonate A to S_A, he has to guess the authenticator Auth1_{AS} and tries to send a corrector one, which is reduced to online guessing the password pw_A against the user since each authenticator sent by the adversary has been computed with at most one password. If the adversary tries to impersonate S_A to A, he has to guess the correct password in order to send such $X^\star$ that he can know x exactly and thus compute $Z_{AS} = xyP$ which is used by A to compute its authenticator Auth1_{AS} and the session key K_{AS}, which is certainly reduced to online guessing the password against the user. Otherwise, the attacker can not know the real value of x(due to the hardness of discrete logarithm problem). One can remark that Y is generated by A in this case and the value of y is also unknown to the attacker. As a result, he can not compute $Z_{AS} = xyP$ and thus will not be able to compute the valid authenticator Auth1_{SA}. Therefore, the user will know it is an illegal server and will halt the processing. If the adversary mounts a passive attack, one can also know that the attacker can not know nothing about session key K_{AS} yet since either x or y is unknown

to him and he can not compute $Z_{AS} = xyP$ either. Therefore, an exhaustive on-line attack is the "best" possible strategy for an attacker. One can invalidate or block the use of a password whenever a certain number of failed attempts occurs. In a word, our protocol is a secure password-based protocol. In addition, with the same analysis, even when pw_A is compromised, the adversary can not know the previous session keys that were established before the corruption (which is usually called forward security) since the session of this type must involve with both legal user and server.

Secondly, the protocol given in Section 3.1.2 for call setup can also provide mutual authentication and establish a secret session key only known by the two communicating users. Based on the above analysis, A(resp. B) will share the secret session keys K_{AS}(resp. K_{BS}) with S_A (resp. S_B) respectively after successful registration. Therefore, each message sent between user and server can be authenticated securely in the protocol. Moreover, replay attack is impossible since a fresh counter value is involved in the computation of authenticators. Therefore, if the two servers are honest, the attacker can not know anything about session key K_{AB} yet since either c or d is unknown to him and he can not compute $Z_{AB} = cdP$ either.

Finally, the scheme given in Section 3.2 can protect the user's personal information and his real identity if the symmetric encryption used is a secure scheme (indistinguishable under an adaptive chosen-ciphertext attack (CCA2) [29]). The intuition behind it is obvious. It directly follows from the definition of security for the symmetric encryption used.

3.4 Performance

Our protocol is efficient. One can easily remark that the communication cost remains unchanged in terms of rounds when compared with previous schemes since the communication flows are also consistent with that of the standard SIP protocol. And the details of comparisons in computation cost between our protocol and the recently proposed EC-based schemes so far as I know are shown in Table 2. Note that we only count the number of point multiplication, pairing operation and public key operation(i.e. encryption/decryption or signature/verificaion), which entail the highest computational complexity, and neglect the computational complexity of all other operations such as Hash computation and symmetric key operation, which can be done efficiently. As shown in Table 2, in one run of our protocol either for registration or for call setup, each participant performs only two point multiplications of elliptic curve but not any pairing computation, which is considered to be too expensive for implementation. Obviously, our scheme is much more efficient than the schemes in [12, 13, 16]. Although the scheme [15] is also quite efficient, it can not be suited for the scenarios where the caller and the callee are in different domains since it assumes that each user must share a secret key with the common authentication center. Moreover it requires a secure device to store the high-entropy key. While the smart card plays an important role in storing sensitive information, it is impractical in many real environments due to inconvenience. In our scheme, the user does not need to carry smart card storing his private information but just needs to know his identity and password. In other words, our scheme is password-only authentication protocol for SIP. To the best of our knowledge, there exists some password-only authentication protocols for SIP in the literature. Unfortunately, as mentioned in introduction all of them are not secure(e.g. [4, 9, 11, 14]. Please note the scheme in [9] can be broken by using an off-line password guessing attack similar to that given in [24].

Table 2 Efficiency comparisons

Schemes	Computation cost*		Password-only
	Registration	Call setup	
Ring's [12]	–	>**2 PM+2PR	No
Wang's [13]	–	>**2 PM+2PR	No
Wu's [15]	2 PM	–	No
Liao's [16]	–	6(5) PM	No
Ours	2 PM	2 PM	Yes

*PM elliptic curve point multiplication; PR Elliptic curve pairing operation.
**The scheme also requires digital signature and verification computation

Based on the results listed in the table, we conclude that our scheme is more practical than the related authentication schemes for SIP.

4 Security proof

In this section, we show that our protocol is secure in the random-oracle(ideal hash function) model, starting with the formal security models and some algorithm assumption that will be used in our proof.

4.1 Security model

In this section, we introduce the formal security models which will be used in next section when we show that our protocol is secure in the random-oracle model. The model builds upon the previous one presented in [30, 31]. In our model, we add one more oracle— *Corrupt*

oracle so that the adversary capabilities in a real attack can be modelled better. Due to the omission of the *Corrupt* query in their model, the protocol proposed by Abdalla and Pointcheval in [31] was found insecure in [32] even if it was provably secure in their model. Moreover, the model is a slightly different variant of that introduced in [31], in which two trusted servers are contained when we consider the call setup phase.

4.1.1 Protocol syntax

Protocol participants and long-lived keys In registration phase, each participant in authenticated key agreement is either a client (User) $U \in \mathcal{U}$ or a trusted server $S \in \mathcal{S}$. Each of them may have several instances called oracles involved in distinct, possibly concurrent, executions of the protocol. We denote U (resp. S) instances by U^i (resp. S^j) or by I when we consider any participant instance. And the client U pre-shares a password pw_U with the server S. In the call setup phase, there involves two clients U_1, U_2 and two corresponding servers S_1, S_2 responsible for their domains respectively. Each client shares a secret key with its home server. And the two servers are connected through a pre-existing secure channel. In this case, we simply denote by U and S when we consider any client and server respectively.

Partner An instances is said to be partner of another instance if it has accepted with the same session identifier SID as the latter's, where SID is defined as the concatenation of all messages an instance has sent and received.

4.1.2 Communication model

The authenticated key agreement protocol $\mathcal{P}$ is an interactive algorithm between U^i and S^j(resp. $U_1^{i_1}, S_1^{j_1}, S_2^{j_2}, U_2^{i_2}$ for call setup phase) that provides the instances of the two communicating parties with a session key *sk*. The interaction between an adversary $\mathcal{A}$ and the protocol participants occurs only via oracle queries, which model the adversary capabilities in a real attack. The types of oracles available to the adversary are as follows:

- *Execute*(U^i, S^j): This query models passive attacks in which the attacker eavesdrops on honest executions between a user instance U^i and a server instance S^j. The output of this query consists of the messages that were exchanged during the honest execution of the protocol. In the call setup phase, the corresponding query is *Execute*$(U_1^{i_1}, S_1^{j_1}, S_2^{j_2}, U_2^{i_2})$ in which an honest execution involves the client instances $U_1^{i_1}$ and $U_2^{i_2}$ and trusted server instances $S_1^{j_1}$ and $S_2^{j_2}$.
- *Send*(U^i, m): This query models an active attack, in which the adversary may intercept a message and then modify it, create a new one, or simply forward it to the intended receiver. The output of this query is the message that user instance U^i would generate upon receipt of message m.
- *Reveal*(I^i): This query models the misuse of the session key by instance I^i (known-key attacks). If a session key is not defined for instance I^i then return $\perp$. Otherwise, return the session key held by the instance I^i.
- *Corrupt*(U): This query returns to the adversary the long-lived key for participant U. As in [33], we assume the weak corruption model in which the internal states of all instances of that user are not returned to the adversary.

Obviously, the adversary is in complete control of every aspect of all communications between participants in the model.

4.1.3 Security definitions

As already noticed, the aim of the adversary is to break (1) the privacy of the session key (a.k.a., semantic security) or (2) the authentication of the players (having a player accepting while no instance facing him). The security notions take place in the context of executing $\mathcal{P}$ in the presence of the adversary $\mathcal{A}$. One first initializes the system parameters, generates a secret key for each user, then provides coin tosses to $\mathcal{A}$, all oracles, and runs the adversary by letting it ask any number of queries as described above, in any order.

Forward security In order to model the forward security (FS) of the session key, we consider the game in which an additional oracle is made available to the adversary: the Test(I^i) oracle. The *Test*−query can be asked at most once by the adversary $\mathcal{A}$ and is only available to $\mathcal{A}$ if the attacked instance I^i is FS-Fresh, which is defined to avoid cases in which adversary can trivially break the security of the scheme. In this setting, we say that a session key *sk* is FS-Fresh if all of the following hold: (1) I^i has accepted, (2) no *Corrupt*-query on I has been asked since the beginning of the game; and (3) no *Reveal*-query has been asked to I^i or to its partner (defined according to the session identification). In other words, the adversary can only ask *Test*-queries to instances which had accepted before

the *Corrupt* query on the related clients is asked. This query is answered as follows:

– $Test(I^i)$:If no session key for instance I^i is defined, then return the undefined symbol $\perp$. Otherwise, flip a (private) coin b and return the session key for instance I^i if $b = 1$ or a random of key of the same size if $b = 0$.

When playing this game, the goal of the adversary is to guess the bit b involved in the *Test*-query, by outputting this guess b'. We denote $Pr[b = b']$ as the probability that $\mathcal{A}$ correctly guesses the value of b. Thus we define $\mathcal{A}$'s **advantage** in breaking the semantic security with regard to $\mathcal{P}$ as $Adv_{\mathcal{P}}^{fs}(\mathcal{A}) = 2Pr[b = b'] - 1$. The protocol $\mathcal{P}$ is said to be (t, ε)-FS-secure if $\mathcal{A}$'s advantage is smaller than ε for any adversary $\mathcal{A}$ running with time t. The definition of time-complexity that we use henceforth is the usual one, which includes the maximum of all execution times in the games defining the security plus the code size [34].

Authentication Another goal is to consider unilateral authentication of either U or S wherein the adversary impersonates a party. We denote by $Succ_{\mathcal{P}}^{im}(\mathcal{A})$ the probability that $\mathcal{A}$ successfully impersonates a U(or S) instance in an execution of $\mathcal{P}$, which means that S(resp. U) agrees on a key, while the latter is shared with no instance of U(resp. S). We say a protocol $\mathcal{P}$ is said to be (t, ε)-Mutual-Authenticated if both $Succ_{\mathcal{P}}^{im}(\mathcal{A})$ is smaller than ε for any adversary $\mathcal{A}$ running with time t.

Usually, we say a protocol is secure if ε can be negligible (in the security parameter l). However, to prevent dictionary attack, ε is just required to be $O(n_{\text{active}}/|\mathcal{D}|) + \epsilon(l)$ for password-based protocols, where $|\mathcal{D}|$ is the size of the dictionary $\mathcal{D}$, n_{active} is the number of active attacks and $\epsilon(l)$ is a negligible function depending on the security parameter l.

4.2 Diffie-Hellman assumptions

In this subsection, we recall the computational assumptions upon which the security of our protocol is based upon. Here we follow the description in [35].

A $(t, \varepsilon) - CDH_{P,\mathbb{G}}$ attacker is a probabilistic machine Δ running in time t such that its success probability $\mathbf{Succ}_{P,\mathbb{G}}^{cdh}(\mathcal{A})$, given random elements xP and yP to output xyP(denoted by $CDH_{P,\mathbb{G}}(xP, yP)$), is greater than ε:

$$\mathbf{Succ}_{P,\mathbb{G}}^{cdh}(\mathcal{A}) = Pr[\Delta(xP, yP) = xyP] \geq \varepsilon.$$

We denote by $\mathbf{Succ}_{P,\mathbb{G}}^{cdh}(t)$ the maximal success probability over every adversaries running within time t. The CDH-Assumption states that $\mathbf{Succ}_{P,\mathbb{G}}^{cdh}(t) \leq \varepsilon$ for any t/ε not too large.

4.3 Security proof

At first, we show the protocol in Section 3.1.1 is mutual authenticated and forward secure in Theorem 1 and Theorem 2 respectively:

Theorem 1 *Let $\mathcal{P}$ describe the proposed authentication scheme associated with these primitives as defined in Fig. 5. Then, for any PPT adversary $\mathcal{A}$, $Succ_{\mathcal{P}}^{im}(\mathcal{A})$ is less than $O(q_s/|\mathcal{D}|) + \epsilon(l)$ under the assumptions that the hash function closely behaves like a random oracle and that the CDH assumption holds in $\mathbb{G}$, where q_s represents the number of Send-queries.*

Proof For an easier analysis, we first exclude some unlikely evens in the game: i.e., collisions on the partial transcripts $(X^\star, Y)$ or on hash values. We can safely do so because the probability that such evens appear is negligible.

At this moment, the sessions in the game can be split in three disjoint sub-cases:

– CaseA: Both X^* and Y have been generated by a real instance of S_A and A respectively. In this case, the adversary can not know the secrets x and y because they are chosen by the client and server respectively. And thus she can not compute $Z_{AS} = CDH_{P,\mathbb{G}}(xP, yP)$ based on the CDH assumption. As a result, she can not validate the guessed password according to the received value Auth1_{AS}, Auth1_{SA} as well as K_{AS}(returned by *Reveal*-query). That is to say, these executions provide no useful information to the adversary at all.
– CaseB: X^* have been generated by a real instance of S_A, but Y has been produced by the adversary via *Send*-query. In this case, we just need to consider those sessions that have been accepted. In order to make a session accepted, the adversary has to send a correct Auth1_{AS} along with Y. Without collusion on hash function, each authenticator sent by the adversary has been computed with at most one PW_A value. Thus we have: $Pr[Succ_{\mathcal{P}}^{im}(\mathcal{A})|\text{CaseB}] \leq \frac{q_s}{|\mathcal{D}|}$.
– CaseC: X^* have been produced by the adversary via *Send*-query, but Y has been generated by a real instance of A. If $\mathcal{A}$ has guessed A's correct password when he sends X^* to the server, she may generate a valid authenticator Auth1_{SA} upon receiving Y from the server. But the probability is less than $\frac{q_s}{|\mathcal{D}|}$. On the other hand, if the adversary

has not guessed A's correct password when she sends X^*, she will not be able to know such $x' \in Z_q$ that $x'P = X^* - PW_A$, where PW_A is computed via the random oracle $\mathcal{G}$ and its discrete logarithm with P as the base is unknown at all. Otherwise, we can use the classical oracle replay technique in [36] to construct an adversary to solve the discrete logarithm problem based on $\mathcal{A}$. It is contradict to hardness of the discrete logarithm problem. With no knowledge of x' or y, she can not compute $Z_{AS} = CDH_{P,\mathbb{G}}(x'P, yP)$ based on the GDH assumption. As a result, she can neither query $\mathcal{H}$ on ("2"$\|S_A\|A\|X^\star\|Y\|PW_A\|AR_A\|Z_{AS}$) to compute Auth1_{SA} nor validate other possible values of pw_A according to the received value Auth1_{AS}. Thus we have: $Pr[Succ^{im}_{\mathcal{P}}(\mathcal{A})|\text{CASEC}] \leq O(\frac{q_s}{|\mathcal{D}|}) + \epsilon(l)$ based on the CDH assumption.

As a consequence, one gets the announced result:

$$Pr[Succ^{im}_{\mathcal{P}}(\mathcal{A})] \leq O\left(\frac{q_s}{|\mathcal{D}|}\right) + \epsilon(l).$$

□

Theorem 2 *Let $\mathcal{P}$ describe the proposed authentication scheme associated with these primitives as defined in Fig. 5. Then, for any PPT adversary $\mathcal{A}$, $Adv^{fs}_{\mathcal{P}}(\mathcal{A})$ is less than $O(q_s/|\mathcal{D}|) + \epsilon(l)$ under the assumptions that the hash function closely behaves like a random oracle and that the CDH assumption holds in $\mathbb{G}$.*

Proof Based on Theorem 1, we just need to consider those executions between the real instances of A and S_A since the adversary can not impersonate any of them. Each session key of this kind must be derived from the value $CDH_{P,\mathbb{G}}(xP, yP)$, where both x and y are generated by an instance of S_A and A respectively. With no knowledge of the involved secrets x and y, the adversary is unable to query $\mathcal{H}$ on ("0"$\|S_A\|A\|X^\star\|Y\|PW_A\|AR_A\|Z_{AS}$) to know the session key because he can not know $CDH_{P,\mathbb{G}}(xP, yP)$ based on the hardness of CDH problem. It is also the case even when later he has compromised the user's password pw_A. Although the adversary may get the session keys in some executions via *Reveal*-query, it will not give any information about the targeted session key for *Test*-query to the adversary because x and y are chosen independently. As a consequence, one easily gets the announced result. □

Next, we will show that the protocol given in Section 3.1.2 for call setup can also provide mutual authentication and establish a secret session key only known by the two communicating users. It directly follows from the following theorem:

Theorem 3 *Let K_{AS} and K_{BS} be secret keys as defined in Fig. 6. Then the protocol described in Fig. 6. is forward-secure and mutual authenticated as long as the hash function closely behaves like a random oracle and the CDH problem is hard in $\mathbb{G}$.*

Proof Based on the above analysis, A(resp. B) will share the secret session keys K_{AS}(resp. K_{BS}) with S after successful registration. Therefore, each message sent between user and server can be authenticated securely in the protocol. Moreover, replay attack is impossible since a fresh counter is involved in the computation of authenticators. Therefore, we just need to consider those executions between the real instances of A, B and S since the adversary can not impersonate any of them. Each session key of this kind must be derived from the value $Z_{AB} = CDH_{P,\mathbb{G}}(cP, dP)$, where c, d are generated by an instance of A and B respectively. With no knowledge of the involved secrets c or d, the adversary(including the honest server S) is unable to query $\mathcal{H}$ on ("0"$\|A\|B\|C\|D\|Z_{AB}$) to know the session key K_{AB} because she can not know Z_{AB} based on CDH assumption. It is also the case even when later he has compromised the user's temporary keys K_{AS} and K_{BS}. Although the adversary may get the session keys in some executions via *Reveal*-query, it will not give any information about the targeted session key for *Test*-query to the adversary because c and d are chosen independently. As a consequence, one easily gets the announced result. □

Finally, based on the above two theorems, we conclude that our proposed scheme can achieve provable security in random oracle.

5 Conclusion

In this paper, we have proposed a secure and practical password authenticated key agreement scheme for SIP using elliptic curve cryptography. Our scheme is simple and efficient. And yet it achieves provable security. Therefore, the end result is more suited to be a candidate for SIP authentication scheme. In addition, we also have provided an extended scheme capable of providing anonymity, privacy, and location privacy.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (No. 61101112) and China Postdoctoral Science Foundation (2011M500775).

References

1. Rosenberg J et al (2002) SIP: Session Initiation Protocol. IETF RFC 3261
2. Handley M et al (1999) SIP: Session Initiation Protocol. IETF RFC 2543
3. International Telecommunications Union (1993) ITU-T Recommendation Q.700: Introduction to CCITT Signalling System 7. Recommendation Q.700. International Telecommunications Union
4. Franks J et al (1999) HTTP authentication: basic and digest access authentication. IETF RFC 2617
5. Stefano S et al (2002) SIP security issues: the SIP authentication procedure and its processing load. IEEE Network 16(16):38–44
6. Geneiatakis D, Dagiuklas T, Kambourakis G, Lambrinoudakis C, Gritzalis S (2006) Survrery of security vulnerabilities in session initial protocol. IEEE Commun Surv Tutor 8(3):68–81
7. Sisalemd D, Kuthan J, Ehlerts S (2006) Denial of service attacks targeting a SIP VoIP infrastructure: stack scenarios and prevention mechanisms. IEEE Network 20(5): 26–31
8. Andreas S, Daniel K and Andreas S (2004) SIP security. Security Group, CH-8401
9. Yoon E, Yoo K, Kim C, Hong Y, Jo M, Chen H (2010) A Secure and efficient SIP authentication scheme for converged VoIP networks. Comput Commun 33(14):1674–1681
10. Vesterinen P (2006) User authentication in SIP. TKK T-110.5290 seminar on Network Security, pp 12–11/12
11. Yang C et al (2005) Secure authentication scheme for session initiation protocol. Comput Secur 24:381–386
12. Ring J,Choo K, Foo E, Looi M (2006) A new authentication mechanism and key agreement protocol for SIP using identity-based cryptography. Proc AusCert R&D Stream pp 61–72
13. Wang F, Zhang Y (2008) A new provably secure authentication and key agreement mechanism for SIP using certificateless public-key cryptography. Comput Commun 31:2142–2149
14. Dimitris G, Costas L (2007) A lightweight protection mechanism against signaling attacks in a SIP-Based VoIP environment. Telecommun Syst 36(4):153–159
15. Wu L et al (2009) A new provably secure authentication and key agreement protocol for SIP using ECC. Comp Stand Inter 31(2):286–291
16. Liao Y, Wang S (2010) A new secure password authenticated key agreement scheme for SIP using self-certified public keys on elliptic curves. Comput Commun 33(3): 372–380
17. Yoon E, Shin Y, Jeon I, Yoo K (2010) Robust mutual authentication with a key agreement scheme for the session initiation protocol. IETE Techn Rev 27(3):203–213
18. Xie Q (2011) A new authenticated key agreement for session initiation protocol. Int J Commun Syst 25(1):47–54. doi:10.1002/dac.1286
19. Rhee et al (2009) A remote user authentication scheme without using smart cards. Comp Stand Inter 31:6–13
20. Shamir A (1984) Identity-based cryptosystem and signature schemes. In: Proc. Crypto 1984. LCNS, vol 196, pp 47–53
21. Al-Riyami S, Paterson K (2003) Certificateless public key cryptography. In: Proc. advances in Cryptology-Asiacrypt'2003. LCNS, vol 2894, pp 452–473
22. Girault M (1991) Self-certified public keys. In: Proc. Eurocrypt'91, pp 491–497
23. Petersen H, Horster P (1997) Self-certified keys: concepts and applications. In: Proc. the third international conference on communications and multimedia security, pp 102–116
24. Boyd C,Montague P,Nguyen K (2001) Elliptic curve based password authenticated key exchange protocols. In: Proc. ACISP 2001, pp 487–501
25. Hankerson D, Menezes A, Vanstone S (2004) Guide to elliptic curve cryptography. Springer, New York, USA
26. Koblitz N(1987) Elliptic curve cryptosystem. Math Comp 48:203–209
27. Kong L et al (2006) A lightweight scheme for securely and reliably locating SIP users. In Proc. IEEE workshop VoIP management and security: 9–17
28. Rosenberg J, Schulzrinne H (2002) Session Initiation Protocol (SIP): locating SIP servers, RFC 3263
29. Phan D, Pointcheval D (2004) About the security of Ciphers. In: Proc. the workshop on selected areas in cryptography 2004. LNCS, vol 3352, pp 185–200
30. Bresson E, Chevassut O, Pointcheval D (2004) New security results on encrypted key exchange. In: Proc. PKC 2004. LNCS vol 2947. Springer, pp 145–158
31. Abdalla M, Pointcheval D (2005) Interactive Diffie-Hellman assumptions with applications to password-based authentication. In: Proc. FC'2005, pp 341–356
32. Choo K, Boyd C, Hitchcock Y (2005) Examining indistinguishability-based proof models for key establishment protocols. In: Proc. ASIACRYPT'2005, pp 585–604
33. Bellare M, Pointcheval D, Rogaway P (2000) Authenticated key exchange secure against dictionary attacks. In: Proc. EUROCRYPT'2000, pp 139–155
34. Abdalla M, Bellare M, Rogaway P (2001) The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Proc. CT-RSA'2001, pp 143–158
35. Abdalla M, Chevassut O, Pointcheval D (2005) One-time verifier-based encrypted key exchange. In: Proc. PKC'2005, pp 47–64
36. Pointcheval D (2005) Provable Security for Public Key Schemes. In: Contemporary cryptology (advanced courses in mathematics—CRM Barcelona), pp 133–189

Shuhua Wu is a lecturer of Networks Engineering Department, Information Engineering University, Zhengzhou, China. Currently, he is a postdoctor at the Department of Computer Science and Engineering, Shanghai Jiaotong University. His research interests include cryptology and communication protocols.

Qiong Pu is a lecturer of Electronics Department, Science Institute, Information Engineering University, Zhengzhou, China. Her research interests include computer communication and networks. Currently, she is working toward her PhD degree in CIMS Center from Tongji University.

Fei Kang an associate professor of Networks Engineering Department, Information Engineering University, Zhengzhou, China. Her research interests include cryptology and information security.