Transactions D:
Computer Science & Engineering and
Electrical Engineering
Vol. 17, No. 1, pp. 25–38

SCIENTIA IRANICA

A Robust and Efficient SIP Authentication Scheme

A. Mohammadi-Nodooshan[1,*], Y. Darmani[1], R. Jalili[2], M. Nourani[3] and M. Sayad Haghighi[1]

Abstract. *The Session Initiation Protocol (SIP), which is becoming the de facto standard for the next-generation VoIP networks, is currently receiving much attention in many aspects. One aspect that was not deeply addressed in the original SIP is its authentication procedure. Apart from its security, an SIP authentication procedure should be efficient. This paper proposes a robust and efficient three-party SIP authentication protocol. In this protocol, the end users are authenticated with the proxy server in their domain using the registrar server. Compared to previous works, our proposed protocol is more efficient and secure. To support our protocol with a formal security proof, its model is constructed using High-Level Protocol Specification Language (HLPSL). The model is verified using the model checking tool, AVISPA, and the result confirms that the protocol is quite safe.*

Keywords: *AVISPA; Formal validation; Proxy server; Registrar server; SIP authentication; Three party authentication; User agent client; VoIP.*

INTRODUCTION

In the next few years, cost saving, integration of voice and data and the creation of new forms of services will push the market toward a fast growth in VoIP technology. As estimated in [1], the global number of civilian VoIP users will reach 197.2 million at the end of 2010, which is 40 times more than those in 2004. As VoIP is a general term used for the transmission of voices over an IP network, it firstly needs a protocol to transmit a digitized voice in the form of packets over an IP network. RTP [2] is the most commonly put-to-work protocol for this purpose. VoIP also employs a protocol to transmit the signaling information for initiating, controlling and terminating sessions between users. Currently, SIP [3] is the main protocol of choice to be addressed for this task. SIP and RTP have been standardized by IETF.

Call Setup in SIP

SIP is an application layer protocol widely employed for controlling two-party or multiparty VoIP sessions. Among different entities defined in SIP, we refer to the following:

- User agent: which acts as the endpoint of a typical session.
- Proxy Server (PS): which forwards requests or responses in an SIP network.
- Registrar Server (RS): which is where the user agents register.

SIP is a request-response protocol. The User agent Client (UC) initiates a request and the User agent Server (US) responds to the request. Each session in SIP is comprised of these requests and their corresponding responses. As depicted in Figure 1, in a typical SIP session, at the first step, UC initiates an "INVITE" request and based on its configuration sends the request to PS or directly to US. PS receives the

1. *Department of Electrical Engineering, K. N. Toosi University of Technology, Tehran, P.O. Box 16315-1355, Iran.*
2. *Department of Computer Engineering, Sharif University of Technology, Tehran, P.O. Box 11155-9517, Iran.*
3. *Department of Electrical Engineering, University of Texas at Dallas, P.O. Box 830688, EC 33, Richardson, TX 75083-0688, USA.*
*. *Corresponding author. E-mail: mohammadi@ee.kntu.ac.ir*

Received 16 January 2009; received in revised form 26 September 2009; accepted 31 October 2009

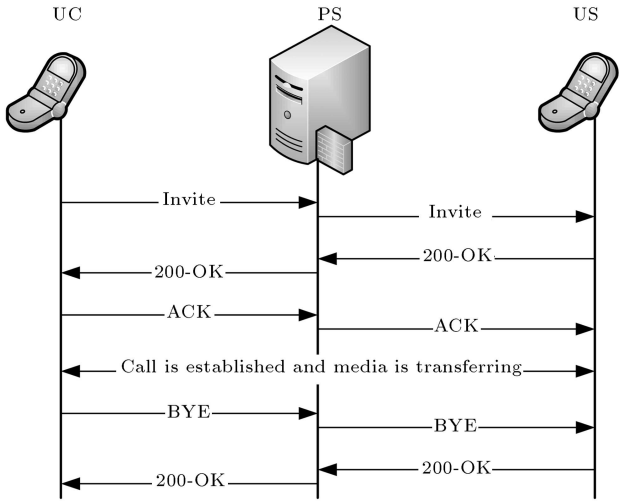

Figure 1. A simple call setup in SIP.

request and forwards it to US. US receives the request and, if it is acceptable, generates a "200-OK" response, which it sends back. PS forwards the received response from US to UC and, then, UC dispatches an "ACK" request to acknowledge the received response.

Authentication in SIP

In a direct session between UC and US, US has to authenticate UC before sending a response to its request. Therefore, in this scenario, SIP needs a two-party authentication protocol. By applying this protocol and based on its SIP configuration UC can prove its identity to US. However, in a proxy-based session, PS has to authenticate UC, using the registration information stored in RS [4].

Over the past century, PSTN has been the dominant telephone network because:

1. It provided high QoS for its users; this is mainly because of the circuit-switched nature of its infrastructure.
2. It provided a good level of security since physical access to the network was quite limited.

Therefore, to replace traditional telephone services, VoIP solutions have to provide a comparable level of performance and security. A pre-requirement in the construction of seamless SIP networks is the designing of secure authentication protocols. A weak authentication protocol may allow unauthorized access to the SIP network, call hijacking, unauthorized call redirection and SPIT, which makes the SIP services unusable. On the other hand, an SIP authentication protocol should be lightweight from computation cost aspects. This is due to the fact that the processing power of the SIP servers is bounded. Therefore, an authentication protocol with a high computation overhead reduces the number of successful calls per second in the SIP network and, also, makes the call setup time unacceptable for end users. An SIP user authentication should not also impose a high communication overhead on the network, since VoIP uses packet-switched IP network as its infrastructure, and any additional communication overhead increases call setup times.

Prior Work

A major problem with SIP security is lack of a strong authentication mechanism. The main SIP authentication mechanism introduced in the SIP RFC is based on the HTTP digest authentication [5]; although, as noted in [6], it does not provide a high level of security. In [7], using state-of-the art verification tool, AVISPA [8], Abdelnur et al. formally verified the authentication procedure in SIP and found that the original SIP authentication can be abused to perform call hijacking and toll fraud. Yang et al. [9] also pointed out that the digest based authentication in the original SIP RFC suffers from off-line password guessing attacks and server spoofing. Based on the Diffie-Hellman (DH) concept, Yang et al. [9] also proposed a new authentication scheme which although resistant against the abovementioned attacks, imposes a large overhead on protocol agents. To moderate this overhead based on the technique employed in [9] and by employing the Elliptic Curve Diffie-Hellman (ECDH) hard problem, Durlanik and Sogukpinar [10] and Wu [11] proposed their authentication and key agreement schemes. To further reduce the authentication overhead of [10], Tsai [12] proposed a nonce-based authentication scheme. In order to avoid the need for PKI or pre-shared passwords, an ID-based SIP authentication and key agreement protocol was proposed by Ring et al. [13]. However, because of the key escrow problem, their protocol could be applied only in a single security domain [14]. To conquer this deficiency, a new SIP authentication and key agreement scheme was proposed in [14], which was based on certificateless public-key cryptography concepts. As claimed by the authors, in all the above works, the authentication process is mutual and resists password guessing (dictionary) attacks. In other words, the attackers cannot verify their guessing of user passwords by any means. This is a mandatory requirement, as in all the above schemes the password is freely chosen by the user and usually these chosen passwords are of low-entropy. However, as recently mentioned by Yoon et al. [15], the protocol presented in [12] is still vulnerable to a password guessing attack.

As opposed to the abovementioned works, Srinivasan et al. [4] targeted the three-party SIP authentication scenario. Their protocol fulfills all the requirements, and its computation overhead in the end

to end communication between UC and US is only 10 milliseconds. As they discuss, this low overhead causes the total call setup time to be well within the acceptable limit recommended by ITU-T [4]. Their scheme uses hash functions, symmetric, private key and public key encryption operations.

Main Contribution

Avoiding the huge computation burden of private key operations, our paper (whose preliminary version was presented in the CSICC-2008 conference) proposes a more secure three-party SIP authentication scheme by employing less private key operations. Also, for prevention of DoS attacks, the proposed scheme moves the unnecessary computation load on the SIP servers to the SIP clients. To resist password guessing attacks, the user's password is selected by the server in the Srinivasan et al.'s scheme. As will be mentioned, this method has a drawback, which we will resolve in our scheme.

Except for [7,11,14], security in the other works is verified informally. However, due to the large complexity of the contemporary security protocols, informal reasoning is not currently sufficient for assuming a protocol to be secure. Therefore, to support our proposed protocol with a formal proof, we model the proposed protocol in the High Level Protocol Specification Language (HLPSL) [16]. The model is verified using the state-of-the art model checking tool, AVISPA, and the result reports the protocol to be a safe one. The results obtained from the verification of a large library of IETF and non-IETF protocols with AVISPA, nominates this tool as a state-of-the art security verification tool [17]. These results show that some attacks are detected by AVISPA that have never been discovered by any previous tools [18].

Authentication Requirements in a Three Party SIP Authentication Protocol

In this section, we mention the requirements, based on which, a three-party SIP authentication scheme is designed. Before running the protocol, just RS and UC share a common secret and during the authentication procedure these goals are satisfied:

G_1: PS authenticates UC.

G_2: If UC is authenticated by PS, PS (using its private key) issues a temporary certificate for UC. As will be discussed, this temporary certificate is required in future authentications between UC and US.

G_3: UC authenticates PS.

G_4: PS authenticates RS.

G_5: RS authenticates UC.

G_6: UC authenticates RS.

G_7: A secret key is established between UC and PS (SK_{UP}), which is used for their further confidentiality and authentication purposes.

G_8: A secret key is established between RS and PS (SK_{RP}), which is used for their further confidentiality and authentication purposes.

G_9: Passwords tables are not stored in RS or PS. This is due to several disadvantages of maintaining password tables (e.g. the risk of modifying the table or impersonating a legal user by stealing the user's password).

G_{10}: It is efficient and does not burden the call setup times [4] (i.e. imposes a low computation overhead on the protocol agents and a low communication overhead on the network).

This paper is organized as follows: The next section reviews the scheme of Srinivasan et al. After that, our authentication scheme is proposed. Then, the following two sections analyze the efficiency and security of the proposed scheme and, finally, in the last section, the conclusion is given.

REVIEW OF THE SCHEME OF SRINIVASAN ET AL.

As our scheme is based on the scheme of Srinivasan et al., this section summarizes their protocol. The notations used throughout this paper are summarized in Table 1.

In the scheme of Srinivasan et al., at the first step, UC registers in RS. During this phase, UC submits I_{UC} to RS, RS generates PW_{UC} and r values, and sends them to UC via a secure channel where:

$$PW_{\mathrm{UC}} = H[N||I_{\mathrm{UC}}], \tag{1}$$

$$r = H[N||I_{\mathrm{RS}}] \oplus H[N||I_{\mathrm{UC}}] \oplus I_{\mathrm{RS}} \oplus I_{\mathrm{UC}}. \tag{2}$$

N is a large number, which is not easy to find by an exhaustive search mechanism. This number is kept secret by RS. Relations 1 and 2 are executed in T_H. Consider that $H[N||I_{\mathrm{RS}}]$ is precomputed in RS, and the overhead of XOR and concatenation operations is negligible.

The second phase is started when UC wants to establish a connection with US and needs to be authenticated to PS. This phase is completed in the following four steps:

Table 1. Notations.

UC	The User Client
US	The User Server
PS	The Proxy Server
RS	The Registrar Server
I_e	Identifier of an entity 'e'
TS_e	Time stamp generated by an entity 'e'
C_e	Certificate of an entity 'e'
PW_{UC}	Password of the user client
$(M)_K$	Message 'M' encrypted using a symmetric key 'K'
$E_K(M)$	Message 'M' encrypted using an asymmetric key 'K'
KR_e	Private key of an entity 'e'
KU_e	Public key of an entity 'e'
$H[M]$	Output digest of a hash function 'H' with 'M' as its input
$e1 \rightarrow e2 : V$	Entity '$e1$' sends vector 'V' to entity '$e2$' via an insecure channel
$e1 \Rightarrow e2 : V$	Entity '$e1$' sends vector 'V' to entity '$e2$' via a secure channel
T_H	Average execution time of a hash function
T_S	Average execution time of a symmetric encryption or decryption operation
T_{PR}	Average execution time of an operation that uses a KR_e for encryption or decryption (e.g. signing)
T_{PU}	Average execution time of an operation that uses a KU_e for encryption or decryption (e.g. signature verification)
$p >>> q$	A strong inequality, in which p is not only greater, but much greater than q
$\|\|$	Concatenation operation
$\oplus$	Bitwise XOR operation

Step 1: $\mathrm{UC} \rightarrow \mathrm{PS} : A$

$$A = n, (R_0)_L, I_{\mathrm{RS}}, TS_{\mathrm{UC}}, \tag{3}$$

$$n = r \oplus PW_{\mathrm{UC}}, \tag{4}$$

$$L = H[PW_{\mathrm{UC}} \oplus TS_{\mathrm{UC}}]. \tag{5}$$

R_0 is a random number generated by UC. Step 1 imposes a computation overhead of $T_H + T_S$ on UC.

Step 2: PS checks TS_{UC} for its freshness, generates a secret random number, σ, and does the following:

$\mathrm{PS} \rightarrow \mathrm{RS} : B$

$$B = \sigma, n, (R_0)_L, TS_{\mathrm{UC}}, \text{Signature of } PS, TS_{\mathrm{PS}}, C_{\mathrm{PS}}, \tag{6}$$

$$\textit{Signature of } PS = E_{KR_{PS}}(H[\sigma, n, (R_0)_L, TS_{\mathrm{UC}}, C_{PS}]). \tag{7}$$

This step imposes an overhead of $T_{PR} + T_H$ on PS.

Step 3: Upon receiving vector B, RS checks whether TS_{PS} is within some elapsed time. Then, it validates C_{PS} and the *Signature of PS*. If these validations are not successful, the processing is terminated. Otherwise, RS finds I_{UC} by:

$$I_{\mathrm{UC}} = I_{\mathrm{RS}} \oplus n \oplus H[N||I_{\mathrm{RS}}],$$

and checks whether UC is legal to make a call. If so, RS computes $L = H[TS_{\rm UC} \oplus H[N||I_{\rm UC}]]$ and deciphers $(R_0)_L$. Then, RS ciphers $E_{KU_{PS}}(H[I_{\rm UC}]||R_0)$ and generates:

Signature of RS $= E_{KR_{\rm RS}}(H[\sigma, \gamma,$

$$E_{KU_{PS}}(H[I_{\rm UC}]||R_0)]), \quad (8)$$

where γ is a secret random number generated by RS. Then:

RS $\rightarrow$ PS : C

$C =\gamma, E_{KU_{PS}}(H[I_{\rm UC}]||R_0),$

Signature of RS, $TS_{RS}, C_{\rm RS}$. (9)

This step needs $T_{PR} + 3T_{PU} + T_S + 6T_H$ of processing time. To investigate this, assume that we use the ITU-T recommendation X.509 with a CA hierarchy height of one level; therefore, validating (the signature part of) $C_{\rm PS}$ is completed in only $T_H + T_{PU}$. Every added level in the CA hierarchy will introduce additional $T_H + T_{PU}$ of processing time to the overhead of the steps of checking a certificate. Note that deriving $KU_{\rm PS}$ from $C_{\rm PS}$ imposes no computation overhead. Note also that validating the Signature of PS takes $T_{PU} + T_H$ of processing time.

Step 4: Upon receiving vector C, PS checks whether $TS_{\rm RS}$ is within the allowed limit. It also verifies the validity of $C_{\rm RS}$ and the *Signature of RS*. If they are valid, PS identifies that UC is an authorized user and issues a temporary certificate ($TC_{\rm UC}$) for UC. Using its private key, PS then decrypts $E_{KU_{\rm PS}}(H[I_{\rm UC}||R_0)$ and computes the session key:

$$SK = H[I_{\rm UC}] \oplus R_0. \quad (10)$$

Then, it sends the value of $(TC_{\rm UC})_{SK}$ to UC. Step 4 has the complexity of $2T_{PR} + 2T_{PU} + T_S + 3T_H$. Note that Issuing $TC_{\rm UC}$ takes $T_H + T_{PR}$ of processing time.

Now, UC computes SK and obtains $TC_{\rm UC}$, which introduces a processing time of T_S (Consider that $H[I_{\rm UC}]$ is precomputed in UC). SK is also used for the encryption of future signaling between PS and UC.

During the call progress period, UC authenticates itself to US, periodically, by means of providing $TC_{\rm UC}, (R_i||TC_{\rm UC})_{SK_i}$ to US where:

$SK_i = H(I_{\rm UC}) \oplus R_{i-1},$

$i = 1, 2, \cdots, n.$ (11)

US verifies $TC_{\rm UC}$, then, computes SK_i to obtain $(R_i||TC_{\rm UC})$ and checks whether the two $TC_{\rm UC}$s are the same. If so, UC is authenticated and US saves R_i to be able to compute SK_{i+1}. As US needs R_0 to be able to compute SK_1, a secure transmission of R_0 from UC or PS to US is needed so that computing SK_1 and the authentication of UC to US, become possible. Encrypting R_0, using the session key between US and PS and sending the encrypted R_0 to US is one solution.

THE PROPOSED SCHEME

Assumptions

Requirement to a PKI

As in our (and the Srinivasan et al.'s) scheme private key operations are employed, the servers need to hold a certificate from a valid CA. Therefore, as the first assumption in our scheme, a PKI is required. However, as described in the "Prior Work" section, some two-party SIP authentication schemes do not need a PKI to be implemented. Compared to these works, our scheme suffers from some of the PKI-based-schemes' drawbacks. First of all, PKI is not widely implemented. Second, it is difficult to manage the revocation mechanism [19]. The other drawback is the heavy computation overhead of processing a certificate which entails public key operations. This, especially, might become an issue in our scheme, where PS and RS have to carry out this operation. However, as mentioned earlier, we can assume that the CA hierarchy level is one in a SIP network. This alleviates the heavy computation overhead on SIP servers .

Trustworthiness of the Servers

In both of the schemes (the Srinivasan et al.'s and ours), we should assume that RS and PS are both trusted. RS should be trusted as US registers in with its private information. PS also have to be trusted, as it forwards UC's requests.

Our proposed scheme is comprised of three protocols, namely registration, login and authentication.

The Registration Protocol (RP)

In this protocol, the following steps are performed:

RP1) UC $\Rightarrow$ RS : $ID_{\rm UC}$.

RP2) RS $\Rightarrow$ UC : $PW_{\rm UC}, C_{\rm RS}$.

$PW_{\rm UC}$ is computed in the same way as in Relation 1. UC has to obtain $C_{\rm RS}$ in any arbitrary way (e.g. receiving it from RS or getting it from a public directory).

PW_{UC} is the shared secret between UC and RS. This will be used by UC to authenticate itself to RS in the next steps. In this way of computing PW_{UC}, there is no need to maintain a password table at the RS side, since PW_{UC} can be derived from I_{UC}. Note that RS uses a single secret, N, for computing the whole set of passwords. Employing this method causes our protocol to satisfy G_9.

The Login Protocol (LP)

This protocol has only one step:

LP1) $\mathrm{UC} \rightarrow \mathrm{PS} : A'$

$$A' = I_{\mathrm{UC}}, authToken, TS_{\mathrm{UC}}, C_{\mathrm{RS}}, \tag{12}$$

$$authToken = H[TS_{\mathrm{UC}}||PW_{\mathrm{UC}}]. \tag{13}$$

UC generates the time stamp, TS_{UC}. Then, by employing its shared secret with the RS (PW_{UC}), it generates a one-time token to authenticate itself to RS. This step partly satisfies G_5.

The Authentication Protocol (AP)

AP authenticates UC and sets a session key between UC and PS (SK_{UP}) along with a session key between RS and PS (SK_{RP}). LP and AP are summarized in Figure 2. AP is completed in the following four phases:

AP-Phase 1

Upon receiving A', PS goes through the following steps:

AP1) PS verifies the freshness of TS_{UC}. This is to check that A' is not a replayed vector.

AP2) PS validates C_{RS} and gets KU_{RS} out of C_{RS}. Note that version 3 of the X.509 specification is used in our scheme, and the certificate issuer utilizes the "Subject" and "Extensions" elements of the certificate to declare the roles which the holder of the certificate can act out in a SIP network; therefore, PS validates C_{RS} to check whether this certificate belongs to a valid registrar server or not.

AP3) PS generates a random number, SK_{RP}, which will become the shared secret between RS and PS. Then, it encrypts TS_{UC}, concatenated with SK_{RP}, using the public key of RS, and com-

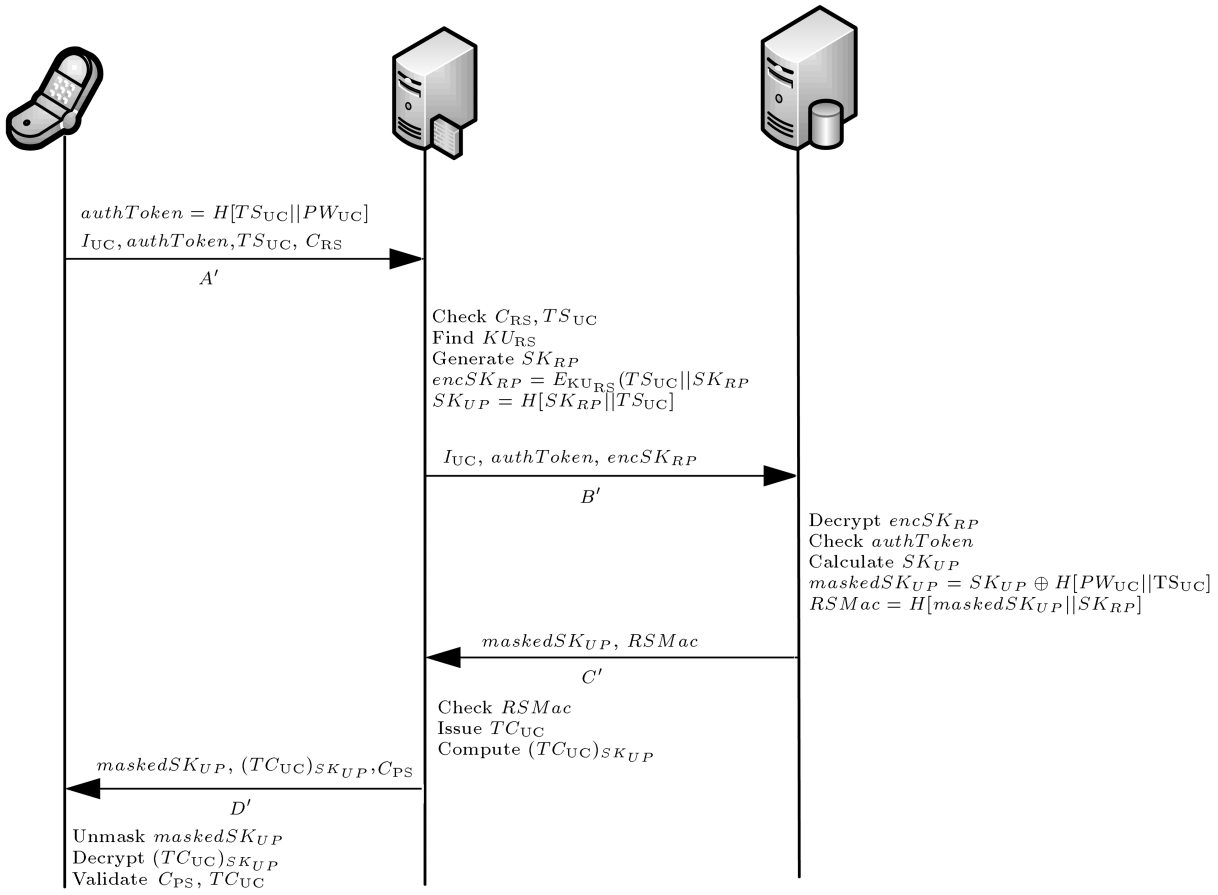

Figure 2. The Login Protocol (LP) and Authentication Protocol (AP).

putes the value:

$$encSK_{RP} = E_{KU_{\mathrm{RS}}}(TS_{UC}||SK_{RP}). \quad (14)$$

This value will be transmitted to RS in AP5. In this step, G_8 is partly satisfied.

AP4) PS computes:

$$SK_{UP} = H[SK_{RP}||TS_{UC}], \quad (15)$$

and saves both SK_{UP} and SK_{RP} for the current session. Upon successful authentication of UC, SK_{UP} will become the shared secret between UC and PS. In this step, G_7 is partly achieved.

AP5) PS $\rightarrow$ RS : B'

$$B' = I_{\mathrm{UC}}, authToken, encSK_{RP}. \quad (16)$$

authToken is sent to RS to authenticate the UC request. Upon a successful authentication, RS uses $encSK_{RP}$ to compute SK_{RP} and SK_{UP}.

AP-Phase 2

Upon receiving B', RS performs the following steps:

AP6) RS decrypts $encSK_{RP}$. TS_{UC} is extracted first to make possible the computation of the value of *authToken* and its comparison with the received one.

AP7) RS checks whether no more than Δt has passed from TS_{UC} to maintain the freshness of the message ($TS_{RS} - TS_{UC} < \Delta t$). One of the goals behind this step is to prevent an attacker from replaying B'.

AP8) RS checks if $H[TS_{\mathrm{UC}}||H[N||I_{\mathrm{UC}}]] = authToken$. If it holds, RS authenticates UC and G_5 is completely satisfied. If UC is legal to make a call, the protocol goes one step forward.

AP9) RS computes SK_{UP} the same way as in Equation 15 and saves SK_{RP} for the current session. Therefore, the common secret between RS and PS is established in this step and G_8 is achieved. SK_{UP} is computed to be sent to UC by RS.

AP10) RS masks SK_{UP} with a "one-time common secret between RS and UC" to compute $maskedSK_{UP}$ as:

$$maskedSK_{UP} = SK_{UP} \oplus H[PW_{\mathrm{UC}}||TS_{\mathrm{UC}}]. \quad (17)$$

The idea behind this step is to send SK_{UP} to UC in a secure, authenticated and efficient manner. The only agent which currently has a shared secret with UC is RS. $H[PW_{\mathrm{UC}}||TS_{\mathrm{UC}}]$ is a one-time common secret between RS and UC and is used to mask SK_{UP}. Therefore, the secrecy of SK_{UP} is preserved as only RS and UC are able to unmask $maskedSK_{UP}$. Therefore, this step partly satisfies G_7. After unmasking this value in AP17, and if SK_{UP} is authenticated in AP19, RS also becomes authenticated to UC and, therefore, G_6 becomes achieved. The masking procedure is also efficient as it only has a computation overhead of T_H.

AP11) RS computes:

$$RSMac = H[maskedSK_{UP}||SK_{RP}]. \quad (18)$$

For the purposes of efficiency, *RSMac* is a three-purpose value. First, it authenticates RS to PS. This is due to the fact that only RS can decrypt $encSK_{RP}$ and extract SK_{RP}. Second, this is a MAC on the value of $maskedSK_{UP}$ as, at this point, the value of SK_{RP} is a common secret between PS and RS. Third, by sending this MAC on the value of $maskedSK_{UP}$, RS informs PS that the UC's request is authenticated. Therefore, this step helps to partly satisfy G_1 and G_4.

AP12) RS $\rightarrow$ PS : C'

$$C' = maskedSK_{UP}, RSMac. \quad (19)$$

AP-Phase 3

Upon receiving C', PS performs the following steps:

AP13) Using the saved value of SK_{RP}, PS verifies $H[maskedSK_{UP}||SK_{RP}] = RSMac$ equality. Based on the discussion in AP11, if this verification is successful, PS authenticates RS and admits to the authenticity of the UC request. Therefore, at this step, G_1 and G_4 are both satisfied.

AP14) As, at the previous step, UC has been authenticated, PS issues a temporary certificate for UC using its private key. Therefore, in this step, G_2 is partly achieved.

AP15) PS computes $(TC_{\mathrm{UC}})_{SK_{UP}}$. The main goal of this step is to send a formatted plain text (TC_{UC}) to UC in an encrypted manner. Employing this method, UC can authenticate SK_{UP}. Authentication of this value ensures UC that there is no MITM between UC and PS.

AP16) PS $\rightarrow$ UC : D'

$$D' = maskedSK_{UP}, (TC_{\mathrm{UC}})_{SK_{UP}}, C_{PS}. \quad (20)$$

AP-Phase 4

Upon receiving D', UC performs the following steps:

AP17) UC calculates $H[PW_{\mathrm{UC}}||TS_{\mathrm{UC}}]$ to unmask SK_{UP} as:

$$SK_{UP} = maskedSK_{UP} \oplus H[PW_{\mathrm{UC}}||TS_{\mathrm{UC}}],$$

therefore, in this step, G_7 is achieved.

AP18) Using SK_{UP}, UC decrypts $(TC_{\mathrm{UC}})_{SK_{UP}}$. At this point, G_2 is satisfied.

AP19) UC validates C_{PS} and checks whether it is a valid proxy server certificate. Then, UC extracts KU_{PS} out of C_{PS}. Using KU_{PS}, PS validates the signed part of TC_{UC} and, if it is valid, PS and RS both become authenticated for UC. If the signature is valid, UC saves TC_{UC} and SK_{UP} for the current session. Otherwise, UC finds out that PS is a bogus proxy server and will terminate any transaction with this proxy server. At this point, G_3 and G_6 goals are achieved.

Through the steps of the protocol, UC and PS have mutually authenticated each other. RS was also authenticated by PS. SK_{UP} has been shared between UC and PS, and SK_{RP} has been shared between RS and PS. Note that SK_{UP} can be employed for future confidentiality and (mutual) authentication purposes between PS and UC. Similarly, SK_{RP} can be used for future confidentiality purposes between RS and PS. However, as the authentication between RS and PS has been unidirectional, SK_{RP} can only be used for authentication of RS for PS. For example, in the confidentiality case, SK_{RP} can be employed in a low-overhead symmetric encryption of $(TS_{\mathrm{UC}}||SK_{RP})$ in step AP3 of another UC authentication. If SK_{RP} is not used for this purpose, the only replacement would be using a heavy asymmetric encryption. However, as mentioned before, SK_{RP} cannot be used for authentication of PS for RS. This is due to the fact that, while sharing SK_{RP} between RS and PS, RS is authenticated by PS, but PS is not authenticated by RS (it is not, in fact, a requirement for the protocol). Therefore, for example, SK_{RP} cannot be used in the encryption of Call Detail Records (CDR) generated at PS and in sending them to RS for logging purposes. This is due to the necessity that the generator of the CDRs should be authenticated by RS, and that SK_{RP} does not convey any information regarding this authenticity.

Table 2 summarizes the steps at which each of the goals is achieved.

During a VoIP conversation, the voices of the end users act as an authentication factor between them. However, if a stricter authentication is required, the procedure used in the scheme of Srinivasan et al., for authenticating the end users to each other, can also be utilized in our scheme. As noted in the "Review of the Srinivasan et al.'s Scheme" section, before running this procedure, the value of R_0 (which is then -during running this procedure- shared between UC and US) should be shared between UC and PS. Therefore, in our scheme, PS and UC will compute the value of $H[SK_{UP}||TS_{\mathrm{UC}}]$ and use it as R_0. Since both SK_{RP} and SK_{UP} are random values, R_0 will be random too. R_0 is also a common secret between PS and UC. Sharing R_0 between UC and PS introduces one additional hash operation in both UC and PS. After sharing R_0, PS will send R_0 to US in a secure manner. Now, UC and US both have R_0, and may use it for authentication purposes.

PERFORMANCE ANALYSIS

Performance Comparison

As our proposed scheme is based on the work of Srinivasan et al., this section reviews the computation and communication performance of the authors' scheme in comparison with the Srinivasan et al.'s.

In the registration phase, both schemes introduce the same computation overhead of T_H. Table 3 summarizes the computation overhead of some of the steps of LP and AP in our scheme.

Based on Table 3, Table 4 compares "the computation overhead of LP and AP on each of the SIP entities" in ours and in the Srinivasan et al.'s scheme. Note that for typical encryption algorithms: $T_{PR} > T_{PU} >>> T_S > T_H$ [20]. Consider, also, that the two additional hash operations required for computing

Table 2. Steps at which each security goal is achieved.

Security Goal	G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8	G_9
Step	AP13	AP18	AP19	AP13	AP8	AP19	AP17	AP9	RP2

Table 3. Computation overhead of some steps of the proposed scheme.

Step	AP2	AP8	AP11	AP13	AP14	AP19
Overhead	$T_{PU} + T_H$	$2T_H$	T_H	T_H	$T_{PR} + T_H$	$2T_{PU} + 2T_H$

Table 4. Comparison of the computation overhead on each of the SIP entities.

Scheme	UC	PS	RS
Srinivasan et al.'s	$2T_S + T_H$	$3T_{PR} + 2T_{PU} + T_S + 4T_H$	$T_{PR} + 3T_{PU} + T_S + 6T_H$
Author's	$2T_{PU} + T_S + 4T_H$	$T_{PR} + 2T_{PU} + T_S + 4T_H$	$T_{PR} + 5T_H$

the value of R_0 in PS and UC in our scheme are also considered in Table 4.

From Table 4, it can be easily recognized that our scheme employs less private-key encryption/decryption operations. This causes the proposed scheme to entail a low computation overhead. The process of authentication between UC and US is the same in both schemes and introduces the same overhead.

Introducing a low communication overhead (total number of bits transmitted) to the VoIP network is one of the other main criteria of any protocol utilized in VoIP environments. It can simply be verified that by utilizing the same encryption functions, the total traffic overload in the VoIP network (introduced for exchanging all vectors A', B', C' and D') in our scheme is less than that of the Srinivasan et al.'s scheme. It should also be noted that both schemes have a minimal number of passes (the number of messages exchanged in a run of the protocol).

Experimental Results

We have implemented each of the protocol agents as a JAVA class. We used the JCA and JCE [21] framework classes for the required cryptographic functions. As these frameworks need an algorithm implementation provider, we employed the Bouncy Castle library [22] as the provider for the frameworks. The algorithms for our public-key, hash and symmetric-key operations were RSA-1024, SHA1 and AES-128, respectively.

Our test-bed was setup in the local area network of Macquarie University. UC, PS and RS were run on three computers, each with a 3GHz CPU and 2GB of RAM. We used the TCP sockets to communicate between these agents. We ran the protocol in a single-threaded mode 20 times and the benchmarks showed 2.1 seconds of computation and communication delay for the whole authentication procedure. We implemented the vectors in four Java classes and used ObjectInputStream and ObjectOutPutStream classes to exchange them. By using such a method, our results showed that a communication overhead of 690 bytes was imposed on the network in our test-bed. It should be noted that the majority of this overhead (72.3%) was imposed on the link between UC and PS and the remaining part was imposed on the link between the servers.

SECURITY ANALYSIS

Formal Security Verification

Designing contemporary security protocols is error prone, and finding their vulnerabilities in an informal manner is too difficult for the human mind. To verify our protocol formally, at the initial step, we modeled it in the HLPSL language. HLPSL is a role-based formal protocol specification language. Therefore, to model the protocol, initially, the three roles of the protocol (UC, PS and RS) were modeled. To accomplish this task, the initial knowledge of the agents and their corresponding state machines were modeled. After modeling the roles of the protocol, at a higher level, a typical session between the roles was modeled and, at the highest level, an environment role was defined. In this environment role, the attacker's knowledge as well as two concurrent sessions was defined. One of these two is a typical session among US, PS and RS, and the other is a session among the attacker, PS and RS, in which the attacker plays the role of UC. As mentioned before, PS and RS are assumed to be trustworthy in a SIP network.

After modeling the protocol in HLPSL, the model was fed to the HLPSL2IF tool of AVISPA, which generated an Intermediate Format (IF) model of the protocol. Then, cl-AtSe back-end of the AVISPA tool was employed to analyze the IF model as this tool supports all operations utilized in our protocol. The final output of the AVISPA tool reported our protocol as being safe, satisfying $G_1 - G_9$. Consider also that G_{10} is an efficiency goal which was reviewed in the "Performance Analysis" section.

Security Discussions

This section discusses six security points of the proposed scheme, denoted as P1 to P6, described below.

P1) In the registration protocol of our scheme, RS does not issue the ID masking number, r, for UC. Instead, UC sends its identity (I_{UC}) to PS in plaintext during the login protocol execution. In the scheme of Srinivasan et al., the r value is used to compute n. To avoid sending I_{UC} in clear text, value n is sent to PS. We argue that it is easy for an eavesdropper to derive I_{UC} from n. To perform this, the eavesdropper "Eve" registers

with RS and gets its own r_{Eve} and PW_{Eve}. Then, Eve simply does the following computation:

$$I_{UC} = n \oplus r_{\text{Eve}} \oplus I_{\text{Eve}} \oplus PW_{\text{Eve}}. \quad (21)$$

Therefore, sending n is as secure as sending I_{UC} in plaintext, and using r and n only imposes some extra overhead on the authentication scheme.

P2) The attendance of TS_{UC} and the random number SK_{RP} in computation of most of the terms in the authentication scheme makes forgery or replay attacks practically infeasible.

P3) Disclosing any of the previous session keys does not affect the security of the new ones. It is also worth mentioning that having the value of SK_{UP}, UC cannot find SK_{RP}.

P4) Knowing the value of R_0, US cannot find SK_{UP}. In the scheme of Srinivasan et al., since US has access to I_{UC} and R_0, it can compute the session key between UC and PS (SK). Having SK, US is able to perform further attacks on the communication line between UC and PS.

P5) Both in our and in the Srinivasan et al.'s protocol, the UC's password is computed as $PW_{\text{UC}} = H[N||I_{\text{UC}}]$. Therefore, the password is not freely chosen by the user and both schemes are resistant to dictionary attacks. However, this method of computing PW_{UC} has a drawback too. The user has to memorize a not-memorable password. To overcome this drawback based on the concepts presented in [23], we propose that users save their passwords on their SIP smart cards. However, if the attacker steals the smart card, user identity becomes revealed and the stealer can masquerade the user. To resolve this issue, the user saves the value of $PW'_{\text{UC}} = PW_{\text{UC}} \oplus PW'$ on their smart-card where PW' is a memorable password chosen by the user. While logging in, the user feeds their SIP phone with the value of PW', the SIP phone computes the value of PW_{UC} as $PW_{\text{UC}} = PW'_{\text{UC}} \oplus PW'$ and uses this value in a regular authentication procedure. This way, if the attacker steals the smart card, he/she cannot find the value of PW_{UC}.

P6) AP7 is designed to prevent an attacker from replaying a valid B' vector and from imposing an extra overhead on RS for the processing of this replayed message. However, as a practical implementation issue, we found that in some cases AP7 has to tolerate some more delay ($\Delta t'$) to receive B', because Δt is affected not only by the network transmission delay, but also by the delay caused by the current computation overhead on the legitimate PS of the domain, which is not exactly predictable. In these cases, an attacker having high computational power can exploit the tolerance gap to pass AP7, but as illustrated (in the formal security verification of the protocol), is not able to breach the security of the protocol. However, replay of B' imposes some overhead on RS, and AP7 is designed to narrow the time window in which the attacker can replay B'.

Evaluation of DoS

As mentioned in [6], SIP is especially prone to DoS attacks, which can be undertaken in several ways including processor overloading. One way to perform this attack is to flood an SIP server with forged messages. According to the SIP RFC [3], without applying a strong authentication protocol in an SIP based VoIP infrastructure, an attacker is able to place spam calls or send manipulated or replayed requests to cause a DoS attack. However, a strong authentication mechanism may itself cause a new form of DoS attack in which the attacker sends numerous bogus messages to the server and depletes those resources allocated for verification of the authenticity of requesters. In this flaw, the attackers may consume a small portion of their system resources to generate bogus messages, while a large overhead is imposed on the SIP servers. Therefore, there is a sophistry here to make a protocol resistant against DoS attacks. We acknowledge that our protocol is prone to DoS attacks and, like other SIP authentication protocols, is not accompanied by a formal DoS-resistant proof. However, in this section we compare our protocol's resistance against DoS attacks with that of the Srinivasan et al.'s work (as it is the only work which has addressed the three-party SIP authentication scenario). To perform this comparison, we take advantage of the concepts presented in [24].

In her work, Meadows proposed a systematic method of proving DoS resistance for a security protocol [24]. However, we adopt the concepts of this paper to make possible a differential DoS-resistance comparison of the two protocols. To investigate the security of a protocol, Meadows compares the cost of the attacker and the cost of the agents of the protocol under attack condition using the "cost sets" concept. To accomplish this, it is required to compute the cost of each event and the chain of actions performed by the attacker on the agents and the attacker itself. Then, based on a tolerance relation defined in [24], a protocol is called DoS-resistant if the final cost of the attacker for performing DoS is large enough compared to the cost of the agents under attack.

On the other hand, to perform an effective DoS attack among many actions and forged messages, attackers choose the one which imposes the most overhead on the server and the least overhead on the machines under their control. This message is

mainly based on the valid messages sent to the SIP server. Therefore, to the end of this section we compare the overhead imposed on the SIP entities to detect forged messages in the worst cases in ours and in the Srinivasan et al.'s scheme. To accomplish this, we first enumerate the main attack scenarios that the proposed scheme encounters to show their cost on the protocol agents. During this enumeration we also show how the proposed scheme resists the attacks. The details of the scenarios are as follows:

Sce1- This scenario is completed in the following steps:

- *Mal* eavesdrops a valid A'.
- *Mal* gets KU_{RS} out of C_{RS}.
- *Mal* performs AP3-AP5.
- RS performs AP6-AP12.
- *Mal* receives C'.
- *Mal* intends to impersonate PS. Therefore, she needs to perform AP14-AP16 to pass AP19, but she cannot, because she has no valid PS certificate or the corresponding private key of a valid PS certificate to issue and sign a valid TC_{UC} for UC. *Mal* also cannot exploit a valid TC_{UC} used in a previous session between UC and PS by eavesdropping D', because TC_{UC} is securely embedded in D'. Therefore, to continue the attack, *Mal* transmits an invalid D' to UC.
- If UC has not received the valid D' of the current session yet, it accepts the invalid D' and performs AP17 and AP18.
- AP19 detects the forgery.

The fourth step of this scenario imposes some overhead on RS. The last two steps also impose some overhead on UC.

Sce2- This scenario is completed in the following steps:

- *Mal* eavesdrops a valid A'.
- *Mal* eavesdrops the valid B' which corresponds to the eavesdropped A' and is sent from PS to RS.
- *Mal* eavesdrops the corresponding C' sent from RS to PS.
- As *Mal* cannot compute SK_{UP} and, based on the discussion in the sixth step of the previous scenario, she cannot perform AP14-AP16, hence, to continue the attack, *Mal* transmits an invalid D' to UC.

The two last steps of the previous scenario are performed.

Sce3- This scenario is completed in the following steps:

- *Mal* eavesdrops a valid A'.
- *Mal* eavesdrops the valid B' which corresponds to the eavesdropped A'.
- *Mal* changes the value of *authToken* in the eavesdropped B'.
- *Mal* sends the forged B' (with the changed value of *authToken*) to RS.
- RS performs AP6-AP7.
- AP8 discovers the forgery.

Sce4- This scenario is completed in the following steps:

- *Mal* eavesdrops a valid A'.
- *Mal* changes the value of *authToken* in the eavesdropped A'. She also updates the value of TS_{UC} to pass AP1.
- *Mal* sends the forged A' to PS.
- PS performs AP1-AP5.

The two last steps of the previous scenario are performed.

Sce5- This scenario is completed in the following steps:

- *Mal* eavesdrops a valid A'.
- *Mal* eavesdrops the valid B', which corresponds to the eavesdropped A'.
- *Mal* changes the value of $encSK_{RP}$ in the eavesdropped B'.
- *Mal* sends the forged B' (with the changed value of $encSK_{RP}$) to RS.
- RS performs AP6-AP12.
- Legal PS of the domain receives C'.
- AP13 detects the forgery.

Sce6- This scenario is completed in the following steps:

- *Mal* eavesdrops a valid A'.
- *Mal* eavesdrops the valid B' that corresponds to the eavesdropped A' and is sent from PS to RS.
- *Mal* eavesdrops the corresponding C'.
- *Mal* changes the value of $maskedSK_{UP}$ or $RSMac$ in the eavesdropped C'.
- *Mal* sends the forged C' to PS.
- AP13 detects the forgery.

Sce7- This scenario is completed in the following steps:

- *Mal* eavesdrops a valid A'.
- *Mal* eavesdrops the valid B' that corresponds to the eavesdropped A' and is sent from PS to RS.
- *Mal* eavesdrops the corresponding C'.

Table 5. The extra overhead imposed on the SIP entities in each of the mentioned attack scenarios.

Scenario	Forged Vectors	Overhead		
		UC	PS	RS
Sce 1	B', D'	$2T_{PU} + T_S + 3T_H$	-	$T_{PR} + 5T_H$
Sce 2	D'	$2T_{PU} + T_S + 3T_H$	-	-
Sce 3	B'	-	-	$T_{PR} + 2T_H$
Sce 4	A'	-	$2T_{PU} + 2T_H$	$T_{PR} + 2T_H$
Sce 5	B'	-	T_H	$T_{PR} + 5T_H$
Sce 6	C'	-	T_H	-
Sce 7	D'	$2T_{PU} + T_S + 3T_H$	-	-

- *Mal* eavesdrops the corresponding D'.
- *Mal* changes some values in D'.
- *Mal* sends the forged D' to UC.
- The last two steps of Sce1 are performed.

Note that in Sce1, Sce3 and Sce5 we assume that the attacker can exploit the tolerance in AP7 and pass this step, but as demonstrated, all the mentioned forgeries are detected by the proposed scheme. However, each attack scenario imposes some extra overhead on the SIP entities to be discovered. Table 5 lists these overheads.

From Table 5, Sce1 and Sce4 impose the largest overheads on the SIP entities. The attacker selects Sce4 because the total overhead imposed on the SIP servers in Sce4 is more than in Sce1. At least 8 other attack scenarios, all similar to the mentioned scenarios, can be enumerated. It can simply be verified that the proposed scheme detects them with equal or less overhead than the mentioned scenarios.

In the scheme of Srinivasan et al., an eavesdropper may dispatch an eavesdropped valid vector A with an updated value of TS_{UC} to PS. This will not be detected before the reception of D by UC and imposes an overhead of $4T_{PR} + 5T_{PU} + 2T_S + 10T_H$ on PS and RS. This is due to the fact that UC is not really authenticated to RS at step 3 in their scheme, hence, this forgery cannot be detected by RS. Note that in the scheme of Srinivasan et al., TS_{UC} and the common secret between RS and UC (PW_{UC}) are used to compute L, R_0 and SK, not to authenticate UC to RS.

Based on Table 5, in the worst cases, in our scheme, overheads of $T_{PR} + 2T_{PU} + 4T_H$ or $T_{PR} + 2T_{PU} + T_S + 8T_H$ are imposed on PS, RS and UC to detect forged messages. It should also be noted that to perform the mentioned DoS in the scheme of Srinivasan et al., the attacker's cost is to change one transmitted value. This cost in our scheme is changing two transmitted values.

CONCLUSION

We demonstrated the demand for a robust and lightweight authentication scheme in SIP networks and outlined the requirements of such an authentication scheme in three-party settings. We also reviewed the scheme of Srinivasan et al., which is the only work that targets all these requirements with an end-to-end overhead within the IETF standard.

Based on the work of Srinivasan et al., we proposed an SIP authentication scheme that complies with all the security goals of a three-party SIP authentication protocol. From a security point of view, this compliance was verified formally using state-of-the art verification tool, AVISPA. It was also discussed how our work enhances the security of the Srinivasan et al.'s method and is more resistant to DoS attacks. From an efficiency point of view, it was analytically illustrated that, compared to the work of Srinivasan et al., our protocol is more efficient in terms of communication and computation costs and reduction in the workload of SIP servers.

To measure the performance of the protocol, the protocol was implemented in JAVA; the experimental results within the setup test-bed demonstrated a total delay of 2.1 seconds and a total communication overhead of 690 bytes. This low overhead makes our scheme a good candidate for use as the SIP authentication protocol in VoIP networks.

REFERENCES

1. "Number of VoIP users shoots up", http://english.peopledaily.com.cn/200509/20/eng20050920_209696.html.
2. Rosenberg, J. et al. "RTP: A transport protocol for real-time applications", *IETF RFC 3550* (2003).
3. Rosenberg, J. et al. "SIP: Session Initiation Protocol", *IETF RFC 3261* (2002).
4. Srinivasan, R. et al. "Authentication of signaling in VoIP applications", *11th IEEE Asia-Pacific Conf. on Commu.*, Perth, Australia, pp. 530-533 (2005).

5. Franks, J. et al. "HTTP authentication: basic and digest access authentication", *IETF RFC 2617* (1999).
6. Salsano, S. et al. "SIP security issues: the SIP authentication procedure and its processing load", *IEEE Netw.*, **16**(6), pp. 38-44 (2002).
7. Abdelnur, H. et al. "Abusing SIP authentication", *J. of Inf. Assur. and Sec.*, **4**(4), pp. 237-242 (2009).
8. "The AVISPA project", http://www.avispa-project.org.
9. Yang, C.C. et al. "Secure authentication scheme for session initiation protocol", *Comp. & Sec. J.*, **24**(5), pp. 381-386 (2005).
10. Durlanik, A. and Sogukpinar, I. "SIP authentication scheme using ECDH", *World Enformatika Society Trans. on Eng. Com. and Tech.*, **8**, pp. 350-353 (2005).
11. Wu, L. "A new provably secure authentication and key agreement protocol for SIP using ECC", *Comp. Standards & Interfaces*, **31**(2), pp. 286-291 (2009).
12. Tsai, J.L. "Efficient nonce-based authentication scheme for session initiation protocol", *Int. J. of Netw. Sec.*, **8**(3), pp. 312-316 (2009).
13. Ring, J. et al. "A new authentication mechanism and key agreement protocol for SIP using identity-based cryptography", *Asia Pacific Inf. Tech. Sec. Conf.*, Gold Coast, Australia, pp. 61-72 (2006).
14. Wang, F. and Zhang, Y. "A new provably secure authentication and key agreement mechanism for SIP using certificateless public-key cryptography", *Comp. Commu.*, **31**(10), pp. 2142-2149 (2008).
15. Yoon, E.J. et al. "A new authentication scheme for session initiation protocol", *Complex, Intelligent and Software Intensive Systems Conf.*, Fukuoka, Japan, pp. 544-559 (2009).
16. Chevalier, Y. et al. "A high level protocol specification language for industrial security sensitive protocols", *Workshop on Specification and Automated Processing of Sec. Requirements (SAPS 2004)*, Linz, Austria (2004).
17. "Deliverable 6.1 of the AVISPA project", http://www.avispa-project.org/delivs/6.1/d6-1/index.html.
18. Armando, A. et al. "The AVISPA tool for the automated validation of internet security protocols and applications", *17th Int. Conf. on Comp. Aided Verification (CAV 2005)*, Edinburgh, Scotland, UK, pp. 281-285 (2005).
19. Paterson, K.G. and Price, G. "A comparison between traditional public key infrastructures and identity-based cryptography", *Inf. Sec. Tech. Rep.*, **8**(3), pp. 57-72 (2003).
20. Menascé, D. "Security performance", *IEEE Internet Computing*, **7**(3), pp. 84-87 (2003).
21. "Java cryptography architecture", http://java.sun.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html.
22. "The bouncy castle project", http://bouncycastle.org.
23. Chien, H.Y. et al. "An efficient and practical solution to remote authentication: smart card", *Comp. & Sec.*, **21**(4), pp. 372-375 (2002).
24. Meadows, C. "A formal framework and evaluation method for network denial of service", *IEEE Comp. Sec. Foundations Workshop*, Mordano, Italy, pp. 4-13 (1999).

BIOGRAPHIES

Alireza Mohammadi-Nodooshan received his BS in Computer Engineering from Isfahan University of Technology in 2005 and his MS in Information Technology from the K. N. Toosi University of Technology, in 2008. His domains of interest include VoIP Signaling Protocols Engineering and Implementation, Computer Security and Bioinformatics in which fields he has also published various papers including the best paper of the CSICC'08 security track.

Yousef Darmani received his BS in Electronics from the Science and Technology University, Tehran, Iran, in 1987, and his MS in Digital Electronics from Sharif University of Technology, Tehran, Iran, in 1991. He received his PhD in Computer Networking from the University of Adelaide, in Australian, in 2004, and subsequently joined the Electrical Engineering Department of K. N. Toosi University of Technology, in Iran, where he is currently working as Assistant Professor. His research interests are VOIP, Real Time Communication over the Internet, Wireless and Ad-hoc Networks and their Protocols and Computer Hardware.

Rasool Jalili was born in Iran in 1961. He received his Bachelor's Degree in Computer Science from Ferdowsi University of Mashhad, in 1985, and his Master's Degree in Computer Engineering from Sharif University of Technology, in 1989. He received his PhD in Computer Science from the University of Sydney, Australia, in 1995, and joined the Department of Computer Engineering at Sharif University of Technology, in Tehran, Iran, where he is now Associate Professor. He has published more than 130 papers in international journals and conference proceedings in the fields of Computer Security and Pervasive Computing. His research interests include such areas as Access Control, Vulnerability Analysis and Database Security, which he conducts at his network security laboratory (NSC, nsc.sharif.edu).

Mehrdad Nourani received his BS and MS degrees in Electrical Engineering from the University of Tehran, in Iran, and his PhD in Computer Engineering from Case Western Reserve University, in Cleveland, Ohio. He worked in the Department of Electrical and Computer Engineering, at the University of Tehran, from

1995 to 1998, and in the Department of Electrical Engineering and Computer Science, at Case Western Reserve University, from 1998 to 1999. Since August 1999, he has been on the faculty of the University of Texas at Dallas, where he is currently an Associate Professor of Electrical Engineering and a member of the Center for Integrated Circuits and Systems (CICS). He is a co-founder of the Quality of Life Technology (QoLT) Laboratory in UTD, where he conducts an interdisciplinary research laboratory focusing on the development of innovative technology and systems to improve people's quality of life.

He has published over 150 papers in journals and refereed conference proceedings, including a best paper award at the 2004 International Conference on Computer Design (ICCD).

Dr. Nourani was recipient of the Clark Foundation Research Initiation Grant in 2001, the National Science Foundation Career Award (2002), and the Cisco Systems Inc. URP Award (2004). His current research interests include Fault-Tolerant Architecture, System-on-Chip Testing, Signal Integrity Modeling and Testing, Application Specific Architecture for Medical Applications and High-Speed Packet Processing Methodologies and Architecture. He is a member of the IEEE Computer Society and the ACM SIGDA.

Mohammad Sayad Haghighi received a BS in Telecommunications from Tehran University, in Iran, in 2001. He then pursued his studies in the area of communication coding and received a MS degree in 2003 from Sharif University of Technology, Tehran. He is currently a PhD candidate in K. N. Toosi University of Technology, in Iran. His research interests cover both Signal/Image Processing and Wireless Network Modeling Topics.