SIP Security Issues: The SIP Authentication Procedure and its Processing Load

Stefano Salsano, DIE — Università di Roma "Tor Vergata"
Luca Veltri, and Donald Papalilo, CoRiTeL — Research Consortium in Telecommunications

Abstract

Session Initiation Protocol is currently receiving much attention and seems to be the most promising candidate as signaling protocol for the current and future IP telephony services, also becoming a real competitor to plain old telephone service. For the realization of such a scenario, there is the obvious need to provide a certain level of quality and security, comparable to that provided by the traditional telephone systems. While the problem of QoS mostly refers to the network layer, the problem of security is strictly related to the signaling mechanisms and the service provisioning model. For this reason, at present, a very hot topic in the SIP and IP telephony standardization track is security support. In this work, the security model used by SIP is described, and the different open issues are highlighted. We focus, in particular, on the problem of authentication providing a short tutorial on the solution under standardization. The architecture of a possible commercial IP telephony service including user authentication is also described. Finally, we focus on performance issues. By means of a real testbed implementation, we provide an experimental performance analysis of the SIP security mechanisms, based on our open source Java implementation of a SIP proxy server. The performance of the server has been compared with and without security support, under various scenarios.

Session Initiation Protocol (SIP) [1] is the Internet Engineering Task Force (IETF) standard for IP telephony. It seems to be the most promising candidate for call setup signaling for future IP-based telephony services, and it has been chosen by the Third-Generation Partnership Project (3GPP) as the protocol for multimedia application in 3G mobile networks.

Within the traditional public switched telephone network (PSTN) a good level of quality of service (QoS) and security has been established over the years, and it is now widely guaranteed. If the new IP telephony architecture and SIP want to replace the PSTN, proposing new service scenarios, they should provide the same basic telephony service with a comparable level of QoS and network security. While the problem of QoS support mainly concerns the IP network layer, the problem of security involves the network layer, the service and control architecture, and its signaling protocols. The following security characteristics should be guaranteed: high service availability, stable and error-free operation, and protection of the user-to-network and user-to-user traffic (for both control and user data).

Although security and privacy should be mandatory for an IP telephony architecture, most of the attention during the initial design of the IETF IP Telephony architecture and its signaling protocol, SIP, has been focused on the possibility of providing new dynamic and powerful services, and simplicity. Less attention has been paid to security features. For this reason, a very hot topic in the SIP and IP telephony standardization track is now how security support can be enhanced to an acceptable level for the type of service that must be provided. In this work these security aspects are considered, focusing on the IP telephony architecture provided by SIP and its extensions for reaching adequate security. Performance cost of a security mechanism is analyzed. A discussion of security aspects in SIP is given; we focus on the authentication procedure. We describe a possible SIP-based telephony service scenario and the related SIP security procedures. Our testbed and methodology for evaluating the processing cost of SIP security are reported with experimental results.

Security Mechanisms in SIP

SIP [1, 2] is an application-layer control protocol that can establish, modify, or terminate user sessions. SIP is a text-based client-server protocol, where the client initiates SIP requests and a server responds to requests. Different types of entities are defined in SIP: user agents, proxy servers, redirect servers, and registrar servers. The user agent represents the terminal (i.e., an application that contains both the user agent client and user agent server). The proxy server is an intermediary entity that acts as both a server and a client for

0890-8044/02/$17.00 © 2002 IEEE

making requests on behalf of other clients. The redirect server accepts requests and replies to the client with a response message (typically providing a contact address for the called user). The registrar is a particular server that accepts user registration requests.

SIP signaling between multiple users consists of requests and responses. When a client initiates a call, an *INVITE* request is sent directly to the IP address of the server or to a locally configured (*outbound*) proxy server. The client sends one or more SIP requests to the server and receives responses from the server. Together, a request and all related responses form a SIP transaction and follow the same signaling path through the same servers. A successful SIP invitation consists of two requests, the *INVITE* message followed by an ACK message. The *INVITE* request asks the callee to join or establish a call. If the callee's response indicates that he/she accepts the call (by sending the *200 OK* response message), the caller confirms that it has received the response by sending the *ACK* message. When the caller or callee wishes to terminate a call, they send a *BYE* request. A reinvitation may be issued during an existing session, in order to change call parameters, by sending another *INVITE* message with the new parameters. SIP messages contain a body that carries the information related to the session to be established, using a text-based representation called Session Description Protocol (SDP).

SIP messages may contain information a user or server wishes to keep private. The headers can reveal information about the communication patterns and content of individuals, or other confidential information. The SIP message body may also contain user information (media type, codec, addresses and ports, etc.) that should not be revealed. Securing SIP header and body information can be motivated by two different reasons:

- Maintain private user and network information in order to guarantee a certain level of privacy
- Avoiding SIP sessions being set up or changed by someone faking the identity of someone else

The mechanisms that provide security in SIP can be classified as end-to-end or hop-by-hop protection. End-to-end mechanisms involve the caller and/or callee SIP user agents and are realized by features of the SIP protocol specifically designed for this purpose (e.g., SIP authentication and SIP message body encryption). Hop-by-hop mechanisms secure the communication between two successive SIP entities in the path of signaling messages. SIP does not provide specific features for hop-by-hop protection and relies on network-level (IPsec [3]) or transport-level (TLS [4]) security. Hop-by-hop mechanisms are needed because intermediate elements may play an active role in SIP processing by reading and/or writing some parts of the SIP messages. End-to-end security cannot apply to these parts of messages that are read/written by intermediate SIP entities.

Two main security mechanisms are used with SIP: authentication and data encryption.

Data authentication is used to authenticate the sender of the message, and to ensure that some critical message information was unmodified in transit. This is to prevent an attacker from modifying and/or replaying SIP requests and responses. SIP makes use of *Proxy-Authenticate*, *Proxy-Authorization*, *Authorization*, and *WWW-Authenticate* header fields, similar to those of HTTP, for authentication of the end system by means of a digital signature. Instead, hop-by-hop authentication can be performed using transport- or network-layer authentication protocols such as TLS or IPsec. Authentication in SIP will be dealt with specifically in the next section.

Data encryption is used to ensure confidentiality of SIP communications, letting only the intended recipient decrypt and read the data. This is usually done using encryption algorithms such as Data Encryption Standard (DES) and Advanced Encryption Standard (AES). SIP supports two forms of encryption: end-to-end and hop-by-hop. End-to-end encryption provides confidentially for all information (some SIP headers and the message body) that does not need to be read by intermediate proxy servers. End-to-end encryption is performed by S/MIME mechanisms (see below). On the contrary, hop-by-hop encryption of whole SIP messages can be used in order to protect the information that should be accessed by intermediate entities, such us *From*, *To*, and *Via* headers. Encryption of such information can prevent malicious users from determining who calls who, or accessing route information. Hop-by-hop encryption can be performed by security mechanisms external to SIP (IPsec or TLS).

SIP messages carry MIME bodies, and the MIME standard includes mechanisms for securing MIME contents to ensure both integrity and confidentiality [5]. As a means of providing some degree of end-to-end authentication, integrity, or confidentiality for SIP header fields, S/MIME can also encapsulate entire SIP messages within MIME. These encapsulated SIP headers are a copy of the "outer" message headers and are used to verify integrity or supply additional private information. Integrity of header fields is performed by matching the value of the header fields in the signed body with that in the outer message. Confidentiality is performed by including in the encrypted body headers that are not present in the outer message. It is important to note that there are some headers that must always have a plaintext version in the outer message (i.e., for all headers used by an intermediate server to route messages). Note also that there may be rare network intermediaries (not typical proxy servers) that rely on viewing or modifying the bodies of SIP messages (especially SDP); secure MIME may prevent these sorts of intermediaries from functioning.

It is worth discussing the above mentioned IPsec and TLS. IPsec is a network layer mechanism that can be used to introduce security directly at the IP layer. Usually IPsec is used to provide security based on network node identity, and this is done independently by the SIP architecture. For this reason, IPsec can be used in SIP mainly between SIP entities that have a preconfigured and quite static security association (e.g., servers within the same IP telephony provider). TLS provides transport-layer security over connection-oriented protocols (TCP), and it is suited to architectures in which hop-by-hop security is required between hosts with a more dynamic security association. Note that if a user agent uses IPsec or TLS to send SIP requests to a proxy server (hop by hop), this does not guarantee that secure transport will be used on the rest of the end-to-end path. The most recent version of the SIP specification [1] includes a way to specify that a resource (e.g., a server or user) should be reached securely using TLS. In particular, the address of a user is normally defined in SIP using a SIP uniform resource identifier (URI) in the form of *sip:bob@biloxi.com*. If a user address is expressed using a new type of URI, a SIP Secure (SIPS) URI (*sips:bob@biloxi.com*), it means that the use of TLS is requested.

The security mechanisms must be combined properly to obtain a trusted network scenario. Just to give an example of this combination, we consider the scenario shown in Fig. 1. User agents authenticate themselves to local outbound proxies using SIP authentication; servers authenticate themselves to other servers one hop away or to user agents with a site cer-

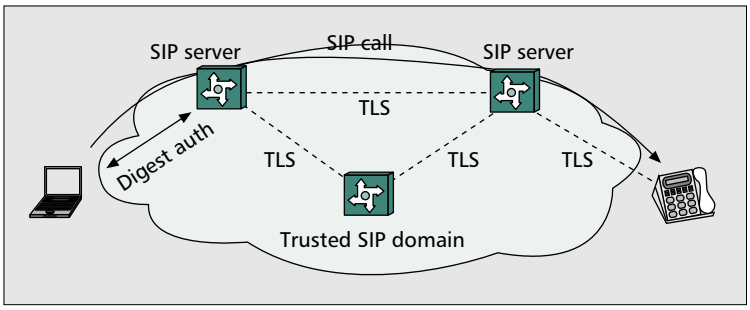

Figure 1. *An example of a trusted network scenario.*

tificate delivered by TLS. In such a way a trusted network architecture can be built, also covering end-to-end SIP paths. Of course this can be complicated if different servers belong to different administrative domains, crossing the so-called trust boundaries. On a peer-to-peer level, user agents ordinarily trust the network to authenticate remote user agents; however, S/MIME can also be used to provide direct authentication between user agents (e.g., if the network is not trusted by the agent).

SIP communications are susceptible to several types of attacks. The simplest attack in SIP is *snooping*, which permits an attacker to gain information on users' identities, services, media, network topology, and so on. This information can be used to perform other type of attacks. *Modification* attacks occur when an attacker intercepts the signaling path and tries to modify SIP messages in order to change some service characteristics. For example, this kind of attack can be used to hijack the signaling flow forcing a particular route, or to change a user registration or modify a service profile. This kind of attack depends on the type of security (or insecurity) used (the type of authentication, number of protected headers, etc.). These attacks can also be used for *denial of service*. *Spoofing* is used to impersonate the identity of a server or user to gain some information provided directly or indirectly by the attacked entity. This attack can be also used to modify a session (e.g., to terminate a call) or to perform *denial of service*. Finally, SIP is especially prone to *denial of service* attacks that can be performed in several ways, and can damage both servers and user agents. The attack techniques may be the same as for other non-SIP systems (e.g., flooding) and may cause memory exhaustion, processor overload, and so on. Further details on possible attacks and protections can be found in [1, 6].

Although the security mechanisms provided with SIP can reduce the risk of attacks, there are some limitations in the scope of the mechanisms that must be considered [1].

The first limitations are with the use of HTTP Digest. First, the integrity mechanisms in Digest do not work very well for SIP since it offers protection only for some SIP parameters, leaving unsigned several header fields user agents might wish to secure. Second, Digest requires that a preexisting secure association can be used in SIP servers where the user is preconfigured. Regarding the use of the S/MIME mechanism, it lacks an infrastructure for user public key exchange. SIP provides a key exchange mechanism [1], but it is susceptible to a man-in-the-middle attack (as are other public-key-based system like SSH). To do so, the attacker has to intercept the first exchange of keys between two parties and remain in the path of all future dialogs. Another difficulty with the S/MIME mechanism is that it can result in very large messages. Finally, regarding the use of TLS, the main problem is that it does not run over UDP and may require maintaining many simultaneous long-lived TLS-over-TCP connections. Note that TLS only allows SIP entities to authenticate servers to which they are adjacent, offering only hop-by-hop security.

Before closing this survey of the security mechanisms in SIP, it is worth considering the current efforts of the standardization process on the improvement of security and trust mechanisms for SIP. An important issue currently under focus is the problem of the agreement on the selected security mechanism between two SIP entities (user agents and/or proxies) that want to communicate enforcing a "sufficient" level of security. As already said, SIP has a number of possible security solutions, some of them directly defined by SIP and others derived by lower protocols (TLS, IPSec, etc.). For this reason, it is very important to define how a SIP entity can select an appropriate mechanism when communicating with a next hop entity. In [7] there is a proposal for a security agreement mechanism that allows two parties to exchange their own security capabilities and preferences in order to select and enforce a common secure framework. In a client-initiated procedure, the SIP agent includes in the first request sent to the next hop entity the list of its supported security mechanisms. The other party (the server side) responds with a list of its own security mechanisms and parameters. The client then selects the highest-preference common security mechanism, turns on the selected security (e.g., a TLS connection), and again contacts the server using the new security mechanism.

Another important issue is the assertion and validation of user identity by SIP servers. The SIP protocol allows a user to assert his/her identity in a number of ways (e.g., in the *From* header); but the identity information claimed by the user is not checked in basic SIP operation. On the other hand, an IP telephony server could need to ensure the identity of a user in order to provide a specific service and/or to condition the type of service to the user's identity itself. The SIP authentication model could be one way to obtain such identity; however, user agents do not always have the necessary key information to authenticate with all other agents. A model is proposed in [8] for "asserted identity" based on the concept of a "trusted domain." The idea is that when a user agent authenticates its own identity with a server, the server may share this authenticated identity (the asserted identity) with all other servers in a trusted domain. A trusted domain is a set of servers that have a mutual preconfigured security association setup. Such security associations represent the trust between the servers. When a server in a trusted domain authenticates the identity of the originator of a message, it adds a new header to the message containing the asserted identity of the user. Such an identity can be used by all other servers belonging to the same trusted domain and then removed (for preserving user privacy) upon exiting such a domain.

The Authentication Procedure in SIP

The authentication mechanism for SIP is described in [1]. Using this mechanism the SIP user agent client (UAC, calling side) is able to identify itself to a user agent server (UAS, called side), to an intermediate proxy server or to a registrar server. Therefore, SIP authentication applies only to user-to-user or user-to-proxy communications; proxy-to-proxy authentication should rely on other mechanisms like IPsec or TLS.

The SIP authentication procedure is derived from HTTP Digest authentication [9]. It 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

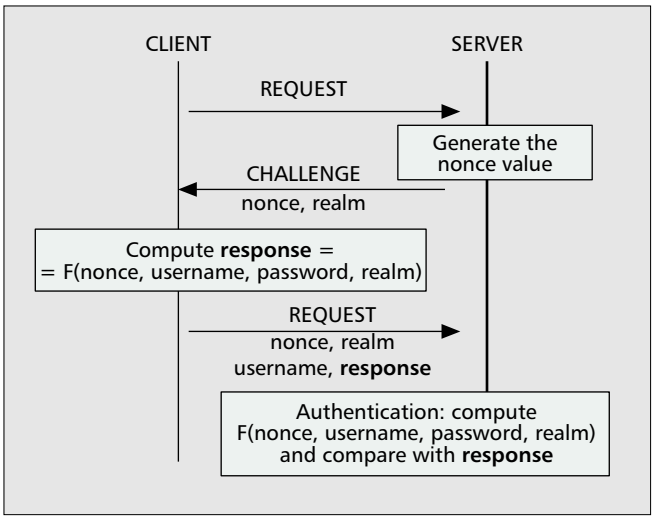

Figure 2. *Digest authentication.*

requester uses this secret password, together with the nonce value, to compute a response value.[1] The requester sends the request again, including the computed response value, which is used by the server to authenticate the request. Using this mechanism, the password is never sent in clear text. A representation of the digest authentication procedure is given in Fig. 2 (slightly simplified with respect to the message parameters). The function *F* used to compute the response specifies how to combine the input parameters with some iterations of a digest algorithm. The specific digest algorithm can be indicated in the challenge, but the default one is MD5 [10]. Note that the server requesting HTTP authentication can be the destination server or an intermediate proxy server: a specific header carried in the challenge allows the user to differentiate the two cases.

The adaptation of this procedure to SIP is straightforward. The authentication procedure is run when the UAS, an intermediate proxy server, or the registrar server requires the calling side (UAC) to be authenticated before accepting the call, forwarding the call, or accepting the registration. The UAS starts sending a "plain" SIP request message (e.g., an *INVITE* to set up a call or a REGISTER to change location information). Upon reception of this message the UAS, proxy server, or registrar server decides that authentication is needed and sends back to the client a specific SIP error message requesting authentication. This error message represents the challenge and includes the nonce and realm. In particular, the error message *401 Unauthorized* is sent by UASs and registrars, while the error message *407 Proxy Authentication Required* is sent by proxy servers. The UAC receives the error message, computes the response, and includes it in a new SIP request message.

Figure 4 shows the message sequence for the case of authentication requested by the proxy server. Note that the UAC sends an *ACK* message immediately after the error message is received. This message closes the first transaction; then the second *INVITE* message opens a new transaction.

The realm parameter, included in the challenge and in the authenticated request, represents the domain within which the user is allowed to receive service. The realm parameter is shown to the user when prompting the user name and password.

Note that the original definition of SIP [2] also allowed a procedure based on HTTP basic authentication [9]. This kind of authentication foresees that username and password are sent in the clear. Basic authentication is subject to packet snooping and therefore is not secure at all. The current SIP specification [1] has deprecated this procedure, and only digest authentication has to be used. Note that digest authentication does not provide a very high level of security by modern security standards. One reason is that it is based on a shared secret rather than a public key mechanism. The SIP protocol also foresees the use of these more secure mechanisms. In particular, [2] described the use of PGP, which could be used a third option (besides basic and digest) in the authentication procedure. The SIP specification in [1] also deprecates the use of PGP and specifies the use of S/MIME to ensure integrity and confidentiality. S/MIME can also provide some kind of end-to-end authentication; for the authentication procedure, only the digest mechanism is described in [1].

An Example Scenario of a SIP-Based IP Telephony Service

It is worth examining in more detail a possible scenario for a SIP-based telephony service in order to analyze the security implications. The service allows users connected to an Internet service provider (ISP) to place calls in the PSTN. The ISP actually uses an Internet telephony service provider (ITSP), which provides the gateway and delivers calls to the PSTN. This scenario has the advantage that the ISP already has a security relationship with its customer. It is "natural" for the ISP to offer this service to the customer in addition to Internet access. A SIP proxy server in the ISP network will be configured as the default outbound proxy server for the SIP clients in the ISP network. This proxy server will forward calls to the ITSP proxy server, which will select and contact the more appropriate SIP gateway (Fig. 3).

In this example scenario, a possible realization of security mechanisms for making calls is as follows. The ISP proxy server uses the proxy authentication procedure of SIP to authenticate the calling user. The ISP is authenticating one of its customers, so it will use already-set-up account information. Existing authentication architectures and protocol (e.g., RADIUS) can be involved on the ISP side in retriev-

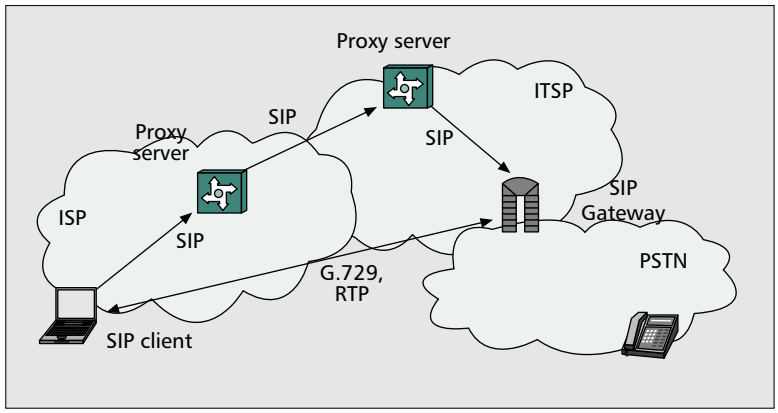

Figure 3. *PC-to-phone call.*

[1] *Actually, the server does not need to know the user password, but rather a digest function of username, realm, and password itself. If an intruder gains this information, he/she will have access to the user resources in this specific realm, but cannot easily derive the password (so user security in other realms is not compromised).*

No authentication, stateless server, UDP ("Basic" call setup procedure)						
Active threads	1	2	3	4	5	6
Single thread throughput (s^{-1})	27.8	15.9	10.6	7.9	6.4	5.3
Total throughput (s^{-1})	27.8	31.7	31.9	31.7	31.8	31.7

Table 1. *Experimental results (threads and throughput).*

ing the customer account information, but this is completely transparent to the SIP procedure. Once the user is authenticated by the proxy server, the proxy checks if the user is authorized to make a call. If this is the case, the proxy contacts the ITSP proxy server by forwarding the SIP *INVITE*. There is no SIP-based mechanism for the ITSP proxy to authenticate the previous proxy, so there are two options for the ITSP proxy: it can request a further proxy authentication of the user or rely on some mechanism external to SIP to authenticate the previous proxy. The first option is clearly unreasonable in this context. The ITSP should replicate the account information for the client, so the first authentication in the ISP would become redundant since the ITSP directly sells the service to the customer. On the contrary, the idea is that the ITSP sees the whole ISP as a customer, so a single-hop authentication between the two proxy servers would fit very well, and TLS or IPsec could be used. Figure 4 provides an example of the resulting call flow for the call setup procedure, showing in particular the simpler possible call flow, including authentication. Next we will discuss some performance issues and describe some alternatives.

Methodology for the Evaluation of Processing Cost and Experimental Results

In order to experiment with advanced features in SIP, we have developed an open source Java SIP server and realized a testbed within CoRiTeL laboratory. The testbed consists of PCs equipped with Linux or Windows 98/2000 operating systems, acting as SIP clients, SIP servers, differentiated services (DiffServ) routers, and DiffServ bandwidth broker. Figure 5 shows the layout of the testbed. In [11], we report on the definition of SIP extensions (QSIP) to support dynamic DiffServ QoS and their implementation in the testbed.

The goal of this testbed was twofold: first, verification of the functional behavior of the various elements and their interoperability; second, the possibility of making some performance analysis. In particular regarding performance analysis, an interesting point is the evaluation of the cost to be paid in terms of performance for the introduction of security mechanisms in SIP, like the authentication procedure. In order to achieve such a goal, we followed a pure experimental approach, trying to evaluate the processing costs of different procedures in the elements of our testbed. Of course, we are conscious that this approach has some limitations, since the results can be severely dependent on the specific characteristics of the implemented modules and the testbed. It is not possible to derive results of general validity; however, we found it useful to have a first estimate of the relative costs of different procedures in the implementation. In particular, we focused on the SIP proxy servers, which are potential bottlenecks in a possible SIP-based telephony service. With this goal in mind, we have defined a methodology to evaluate the processing cost of SIP procedures in the proxy server. The idea is to measure the maximum performance (throughput in terms of procedures per time unit) our implementation of a proxy server can achieve. Then the processing cost of the procedure is derived in inverse proportion to the performance.

Figure 6 shows the test environment used to test the performance of the proxy server. With respect to the complete testbed scenario shown in Fig. 5, all the functionality and components related to QoS have been removed. The "tester" UAC on the left is a multithreaded Java application that generates SIP calls and evaluates call throughput. The proxy server under test performs the "real" processing of the call, while the "tester" UAS is a simple Java application that gives predefined answers according to the correct SIP message flow. Each of the three elements runs in separate PCs (300 MHz Pentium/128 Mbytes RAM, where no other applications are running), and the three PCs are connected to a dedicated Fast Ethernet switch. Each thread in the tester UAC (up to 6) generates a bunch (e.g., 100 in our tests) of SIP calls, one after the other. A new call setup is generated immediately when the previous call setup is completed (with reception of a *200 OK* message or sending of an *ACK* message). The goal is to saturate the processing capability of the server and measure its maximum throughput. More than one thread in the tester UAC is needed for this purpose.

Starting from the SIP-based telephony service scenario described in the previous section, eight procedures/scenarios have been considered in order to compare their processing cost. As reported in Tables 1 and 2, the basic procedure/scenario (1) is a SIP call setup with no authentication, where the proxy server is call stateless and always uses UDP communication. Procedure/sce-

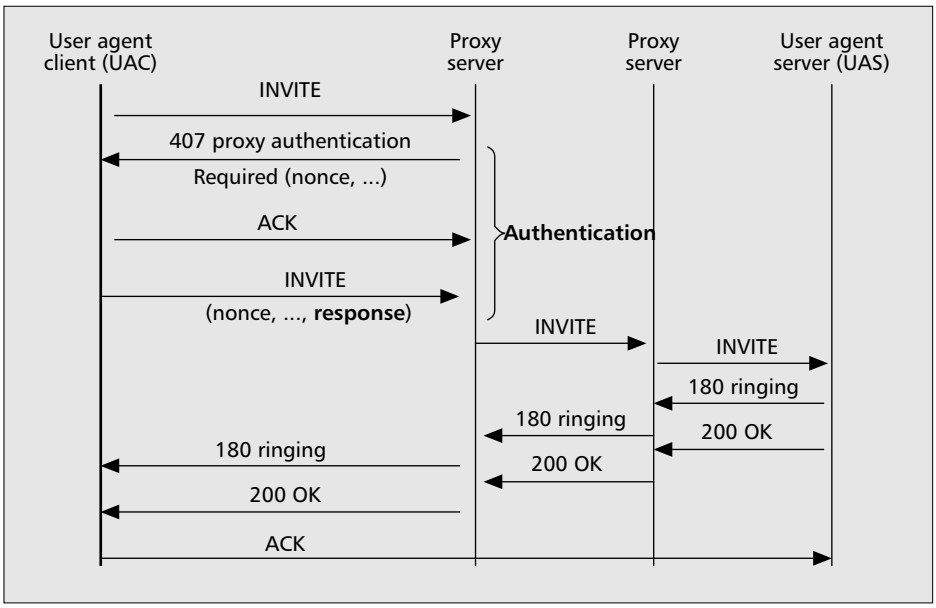

Figure 4. *SIP call setup with proxy and authentication (stateless proxies).*

nario (2) includes authentication and corresponds exactly to the call flow reported in Fig. 4. If we have authentication, we evaluated the use of TLS for securing the communication between the proxy server and the tester UAS (procedure/scenario [4]). Scenario (3) has been considered in order to see the difference between the use of UDP or TCP between the proxy and the tester UAS. The motivation of considering TCP-based SIP communication is to have an incremental analysis toward a TLS-based SIP communication scenario. Procedures/scenarios (5–8) replicates (1–4) as far as authentication and UDP/TCP/TLS are concerned, by considering a call stateful proxy server. In these procedures/scenarios the proxy server receives and processes the *ACK* message from the calling UAC and the *BYE*/*OK* messages that terminate a call. It also keeps the state of active calls and deletes this state when calls are terminated.

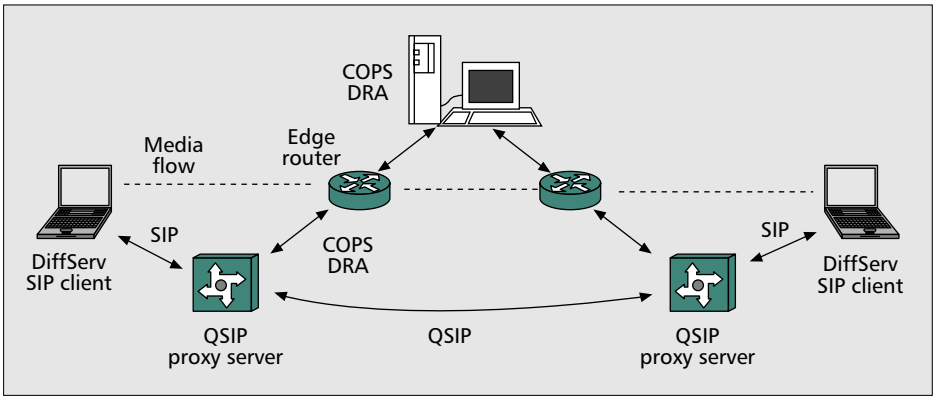

Figure 5. *The complete QSIP testbed scenario in CoRiTeL.*

Several measurements were run to define and validate the test methodology. For example, the performance dependence on the number of pending authentication states or active call states (5–8) has been studied. Thanks to the fact that an hash function has been used in the implementation to operate with authentication and call states for the lookup, insert and delete operations, the processing cost is basically independent of the number of active states. Another aspect that has been considered and tuned is the number of active threads in the tester UAC. Table 1 reports the call throughput for the procedure/scenario (1) for different numbers of active threads. Starting from two threads in parallel, the capacity of the server is saturated: each of the N threads takes $1/N$ of the server capacity, so its throughput is $1/N$ of the maximum throughput to be evaluated. Further details on the measurements, including the complete set of call flows can be found in [12].

In Table 2 the results of our evaluation are reported. The third column reports the theoretical maximum throughput in terms of calls per second the proxy server can handle. This includes all the processing that the proxy performs for a call from its setup to its termination. Note that this throughput

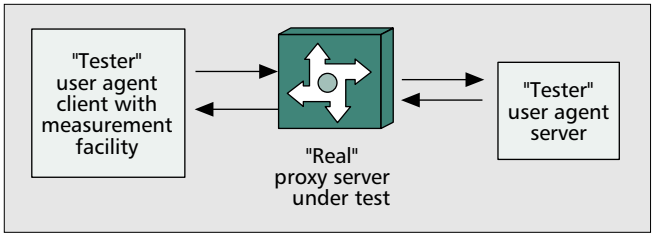

Figure 6. *Testing SIP server performances.*

corresponds to 100 percent utilization of server processing resources; therefore, it is not the operational call rate to which the server should be exposed. The two rightmost columns are the most important ones and report the throughput value converted in a relative processing cost. In the first one the reference value of 100 has been assigned to procedure/scenario (1), while in the last one it has been assigned to procedure/scenario (5).

The results show that the introduction of SIP security accounts for nearly 80 percent of processing cost of a stateless server and 45 percent of a call stateful server. This increase can be explained with the increase in the number of exchanged SIP messages and with the actual processing cost of security (including the cryptographic algorithms). We have estimated that 70 percent of the additional cost is for message processing and 30 percent for actual security mechanisms.

Another interesting finding is that the TCP processing introduces a small increase with respect to UDP and that the additional increase due to TLS is almost negligible. Note, however, that these results have been obtained considering that both the TCP and TLS connections are already established and available when SIP messages need to be sent. This can be reasonable in our scenario, where a relatively stable connection can be assumed between the ISP and ITSP proxy servers. The result cannot be extended in general to the use of TSL between arbitrary SIP entities.

	Procedure/scenario	Total throughput (s^{-1})	Relative processing cost	
1	No authentication, stateless server, UDP ("basic" call setup procedure)	34.8	**100**	
2	Authentication, stateless server, UDP	19.6	177	
3	Authentication, stateless server, TCP	19.4	180	
4	Authentication, stateless server, TLS	19.2	182	
5	No authentication, call stateful server UDP	21.0	166	**100**
6	Authentication, call stateful server, UDP	14.6	239	144
7	Authentication, call stateful server, TCP	13.8	253	152
8	Authentication, call stateful server, TLS	13.6	256	154

Table 2. *Experimental results.*

Conclusions

In this article the main security aspects related to SIP-based IP telephony are discussed. The authentication procedure, based on HTTP Digest authentication, is described. The mapping of authentication mechanisms into a possible SIP-based service scenario are provided. Finally, the performance aspects of SIP authentication are considered with a pure experimental approach. The processing costs of different security proce-

dures/scenarios are compared under a reference implementation. Although the performance results are obviously conditioned by the specific implementation aspects, they can provide a rough idea of the relative processing cost of SIP security procedures.

References

[1] J. Rosenberg *et al.*, "SIP: Session Initiation Protocol," IETF RFC 3261, June 2002.
[2] M. Handley *et al.*, "SIP: Session Initiation Protocol," IETF RFC 2543, Mar. 1999.
[3] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," IETF RFC 2401, Nov. 1998.
[4] T. Dierks and C. Allen, "The TLS Protocol Version 1.0," IETF RFC 2246, Jan. 1999.
[5] B. Ramsdell, Ed, "S/MIME version 3 message specification," IETF RFC 2633, June 1999.
[6] M. Thomas, "SIP Security Requirements," IETF Internet draft <draft-thomas-sip-sec-req-00.txt>, Nov. 2001, work in progress.
[7] J. Arkko *et al.*, "Security Mechanism Agreement for SIP Sessions," <draft-ietf-sip-sec-agree-04.txt>, June 2002
[8] C. Jennings, J. Peterson, and M. Watson, "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks," < draft-ietf-sip-asserted-identity-01>, June 2002
[9] J. Franks *et al.*, "HTTP Authentication: Basic and Digest Access Authentication," IETF RFC 2617, June 1999.
[10] R. Rivest, "The MD5 Message-digest Agorithm," IETF RFC 1321, Apr. 1992.
[11] S. Salsano and L. Veltri, " QoS Control by Means of COPS to Support SIP Based Applications," *IEEE Net.*, Mar./Apr. 2002.
[12] D. Papalilo, S. Salsano, and L. Veltri, "Performance Measurements on the SIP Server — v1." http://www.coritel.it/projects/qsip

Biographies

STEFANO SALSANO (stefano.salsano@uniroma2.it) received his Laurea degree in 1994 (University of Rome "Tor Vergata") and his Ph.D. in 1998 (University of Rome "La Sapienza"). From November 2000 he has been an assistant professor at the University of Rome "Tor Vergata." From 1997 to 2000 he has been with CoRiTeL, a research institute on telecommunications, where he has been coordinator of IP-related research activities. He has participated in several research projects founded by the EU (INSIGNIA, ELISA, AQUILA), the European Space Agency (ESA), and the Italian Ministry of Research. His current research interests include QoS and traffic engineering in IP networks, IP telephony, MPLS, and IP over optical networks.

LUCA VELTRI (veltri@coritel.it) received his Laurea degree in telecommunication engineering from University of Rome "La Sapienza" in 1994. He received a Ph.D. in communication and computer science from the same university in 1999. Since 1999 he has been with CoRiTeL that joins Ericsson Lab Italy and the university. In CoRiTeL he is coordinator of research activities in IP networking. He has participated in several research projects founded by the EU, ESA, and, the Italian Ministry of Research. His current research interests include QoS in IP networks, IP telephony, security, and mobility management in next-generation networks.

DONALD PAPALILO (papalilo@coritel.it) received his Laurea degree in telecommunication engineering from University of Rome "La Sapienza" in 2001. In 2001 he participated in the ESA project LLTP (Transport Protocols and Resource Management for Mobile Satellite Networks). In 2002 he received a scholarship from CNIT (National Inter-University Research Consortium on Telecommunications). His current research interests include QoS in IP networks and IP telephony.