Master's Project Report

Study of Digest Authentication for Session Initiation Protocol (SIP)

Author: Qi Qiu (Student No.: 2453712)

Supervisor: Robert L. Probert

SITE, University of Ottawa

December 2003

Executive Summary

This report is written to fulfill the requirements of the directed study course CSI5901. The area under study is the security for Voice Over IP (VoIP), and in particular the Digest authentication for Session Initiation Protocol (SIP).

SIP authentication scheme borrows HTTP digest authentication algorithm that offers challenge-based authentication and limited integrity protection. This report investigates the Digest authentication scheme for SIP; addresses the issues of applying the Digest authentication mechanism to SIP; proposes extensions to the existing digest authentication scheme. The existing Digest authentication mechanisms and proposed extensions aim to form a robust authentication scheme for SIP-based system.

Specifically, the following aspects are studied:

- Identifying possible security threats and attacks that may exploit the vulnerabilities existed in the authentication functionality. *[Section 2]*
- Reviewing the digest authentication mechanism that is standardized and implemented in SIP. *[Section 3]*
- Evaluating the Digest authentication scheme against the identified threats and attacks. *[Section 3]*
- Proposing an authentication mechanism that can be an extension of the current SIP authentication scheme. *[Section 4]*
- Analyzing the limitations of the authentication scheme and providing a pointer for the future study. *[Section 5]*

Table of Contents

1. Introduction to SIP

In IP telephony, the main complexity of IP telephony lies in signalling process, often referring to call setup and call management [13]. There are two main standards for signalling. The first one is H.323 protocol suite proposed by the International Telecommunication Union (ITU). The second one is the Session Initiation Protocol (SIP) proposed by the Internet Engineering Task Force (IETF). Both protocols possess its own strengths in different aspects [14], but they provide similar mechanisms for call establishment, call control, teardown and supplementary services. SIP is gaining increasing popularity due to its flexibility.

SIP is an application-layer control protocol that can establish, modify, or terminate user sessions. These sessions can include Internet telephony, multimedia conferences, distance learning, multimedia distribution and instant messaging applications, as well as many others.

To better understand SIP authentication, it is important to be familiar with the following SIP concepts and terminologies. A complete description of SIP can be found in [1].

1.1 SIP Participants

SIP uses a client-server model, where the client initiates SIP requests and the server responds to requests. A SIP-based system is usually made up of the following elements:

> **User Agent**: user agent (UA) is an application that contains both the user agent client (UAC) and the user agent server (UAS). UAC places outgoing calls; UAS handles incoming calls.
>
> **Proxy Server**: proxy server (PS) is an intermediary program that forwards requests from user agents to another locations. PS also provides routing, authentication, billing functions, and etc.
>
> **Registrar Server**: registrar accepts the registration requests from user agents. Registrar server is also responsible for authenticating the registration request.
>
> **Redirect Server**: The redirect server accepts SIP requests and maps the address to zero or more new addresses and returns these to the client.

There are also other SIP elements, such as Location Server and DNS, not directly associated with authentication; thus we omitted them from this document.

1.2 SIP Messages

SIP is a text-based protocol that is based on the HTTP protocol [15]. The message syntax and header fields are similar to those of HTTP/1.1, however it should be noted that SIP is not an extension of HTTP. As mentioned earlier, a SIP message is either a request from a client to a server, or a response to a request from a server to a client. A message consists of a start line, one or more header fields, an empty line and an optional message body.

1.2.1 Request Messages

A request message begins with a SIP method name. The method name identifies the type of request and is case-sensitive. Table 1 shows the standard SIP method names. There have also been a number methods proposed for supporting additional features and applications (e.g. MESSAGE for instant messaging). Methods that are not supported by a proxy or redirect server are treated as OPTIONS and forwarded accordingly.

Method	Purpose	Supports Authentication
INVITE	Initiate a session	Yes
ACK	Acknowledge session initiation	No
OPTIONS	Query server capabilities	No
BYE	Terminate a session	Yes
CANCEL	Cancel a pending request	Yes
REGISTER	Register a user’s location	Yes

Table 1: SIP Methods

1.2.2 Response Messages

The response message contains a status code that indicates the result of the server’s attempt to process the request. Many of the SIP response codes are identical to the HTTP/1.1 response codes. SIP response codes are extensible and SIP applications are not required to understand all codes as long the class of the code is recognized. Table 2 shows the status code classes. There two types of status codes, non-final and final. Non-final codes (1xx) are for informational purposes and indicate that the request is being processed but may take some time. All other codes are final codes and represent a conclusion to the transaction. A non-final response code is always followed by a final code.

Status Code	Description	Example
1xx	Informational	100 Trying
2xx	Success	200 OK
3xx	Redirection	300 Multiple choices
4xx	Client error	401 Unauthorized
5xx	Server error	503 Service unavailable
6xx	Global failure	600 Busy everywhere

Table 2: SIP Status Codes

1.3 SIP Functionality

End users are identified by SIP URLs. A SIP URL or address takes the form sip:user@host that is similar to E-mail address. The host part is either a numeric network address or domain name, and the user part is usually a user name.

When a user agent client initiates a call, a request is sent directly to the IP address of the server, or to a locally configured SIP proxy server (known as outbound proxy). Standard name resolution mechanisms (e.g. DNS) are used to determine the address of a server. After the desired server has been located, the client sends one or more SIP requests to the server and receives responses from the server. Together, a request and all responses triggered by it, form a SIP transaction.

SIP signaling between multiple users consists of requests and responses. The most common SIP operation is the invitation. A successful SIP invitation consists of two requests, the INVITE message followed by an ACK message. The INVITE request asks the callee to establish a call. If the callee's response indicates that it accepts the call, the caller confirms that it has received the response by sending the ACK message. If one of users wants to hang up the phone, it sends a BYE request.

A callee may move between different locations, which can be dynamically registered with the location server. A user may also be registered in multiple locations. A SIP redirect server returns a list of locations to the caller as Contact headers. A SIP proxy or server can try these addresses, either sequentially or in parallel, until the call is accepted or declined. If a proxy forwards a request, it adds itself to the list of forwarders in the Via header (Proxy does nothing more than this). This is to ensure that the reply will take the same path back. On the return path, the proxy removes its Via entry, in order to hide the routing information. A client uses the REGISTER request in order to let the servers know where it can be reached.

SIP makes minimal assumptions about underlying transport and network layer protocols. It does not require a reliable transport protocol, but instead provides its own reliability mechanism. In the Internet, SIP normally uses the UDP or TCP protocols, but it can also directly use other protocols such as ATM, IPX, X.25 or frame relay [1].

2. Security Threats and Attacks

SIP system is deployed in the Internet that can be considered hostile environment, in which SIP elements and messages may be exposed to a variety of security threats and attacks. "A threat is, by definition, a vulnerability available to a motivated and capable adversary" [9].

This section presents and analyses some threats that could be used to exploit the SIP authentication aspect of security. The threats and attacks attempting to breach the lower layer encryption protection (e.g. TLS and IPSec [4][11]) are not discussed in this document.

2.1 Replay Attack

Replay attack is that a malicious user retransmits a genuine message in order to establish authorized communication with the entity receiving the message. Replay attack is a common threat to the client-server systems that use messages as communication means. The examples of this type of system are HTTP, SMTP, and SIP.

Among various threats in the Internet-based system, replay attack is relatively easy to launch. [16] illustrates several forms of replay attacks described below:

- Simple replay: the attacker simply eavesdrop a legitimate message and replays it later. Let's look at a simple replay attack: After a protocol session of an sender A and a receiver B, an attacker C captures all the messages sent in the session. Then, C tries to re-send the messages to B to impersonate as A. If C can trick B to finish its session making B believe it is talking to A, then the attack is successful.

- Repetition that can be logged: the attacker replays a timestamped message within a valid time interval.

- Repetition that cannot be detected: the attacker captures a message that does not reach the destination. When the attacker replays the message the receiver accepts it as valid and it cannot detect that another authorized entity has sent it before.

2.2 Registration Hijacking

The SIP registration mechanism provides for a UA to create a binding in a location service for a particular domain that associates an address-of-record with one or more contact addresses. The registrar takes the From address as the asserted identity of the originator of the request. The From header field can be modified by the owner of a UA, which opens the door to malicious registrations. An attacker that successfully impersonates a party can change contacts associated with an address-of-record could, for example, de-register all existing contacts for a URI and then register their own device as the appropriate contact address, thereby directing all requests for the affected user to the attacker's device.

2.3 Request Spoofing

Request Spoofing is used to impersonate the identity of a message sender to fool the legitimate recipient. By changing the message header or body, the malicious person can send a forged request that makes the recipient believe that he/she is communicating with another entity.

There exist three common forms of this type of attack to a SIP request: spoofing INVITE message, spoofing BYE message, and spoofing CANCEL message.

- Spoofing INVITE message

  A common spoofing attack to INVITE is that the attacker spoof the From, Via or Subject header fields. The attacker does not want to use his real address for various reasons (e.g. cheating or billing).

  An example of spoofing the From, Via and Subject header fields in a INVITE request is given below:

  ```
  INVITE sip:bob@biloxi.com SIP/2.0
  Via: SIP/2.0/UDP somewhere.com
  To: Bob <sip:bob@biloxi.com>
  From: Alice <alice@atlanta.com>
  Call-ID: a72bcf72f22d
  Cseq: 100 INVITE
  Contact: <sip:John@somewhere.com>
  Subject: Your Friend Alice
  …
  ```

  In this example, firstly, John is the attacker who wants Bob to believe that the message is from Alice, therefore he puts Alice's address in the From header field instead of his. Secondly, since John's intension is to establish a dialog with Bob with a fake identity, which means that John wants Bob to send the response back to him rather than Alice, John has to specify his own IP address or domain name in Via header field. (If John is a more advanced attacker, he could be able to spoof the IP address to make the trace more difficult or even impossible).

- Spoofing BYE Request

  The purpose of this attack is to tear down a normal conversion session between the legitimate users. To launch this attack, the attacker has to somehow learn the parameters (i.e. From, To, Call-ID, Cseq, etc.) of the previous call control requests, then build his own BYE message by using those captured parameters, and finally send this forged BYE request to the target user.

- Spoofing CANCEL Request

  The purpose of this attack is to stop a legitimate INVITE request by sending a forged CANCEL message to the legitimate callee (Figure 1), so that the legitimate INVITE message will be treated as an invalid request.

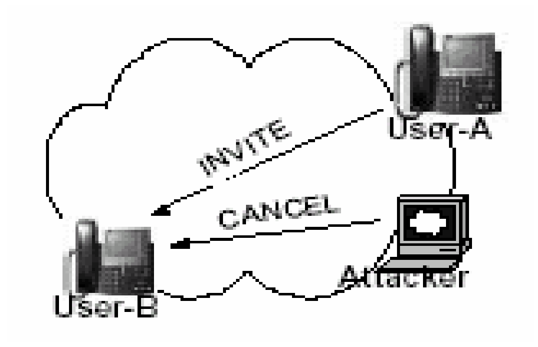


Figure 1. CANCEL Spoofing Attack

2.4 Impersonating a Server

The threats described in section 2.1 to 2.3 are all launched from the client, that is, the attacker impersonates a client. In the meantime, it is also possible for an attacker to impersonate a server and intercept the UA's requests.

For example (Figure 2), a redirect server in a domain biloxi.com impersonates a redirect server in another domain biloxi.com. A UA sends a request to biloxi.com, but the redirect server in biloxi.com responds with a forged message that has appropriate SIP header fields for a response from biloxi.com. The forged contact addresses in the redirection response could direct the originating UA to an insecure entity [1].

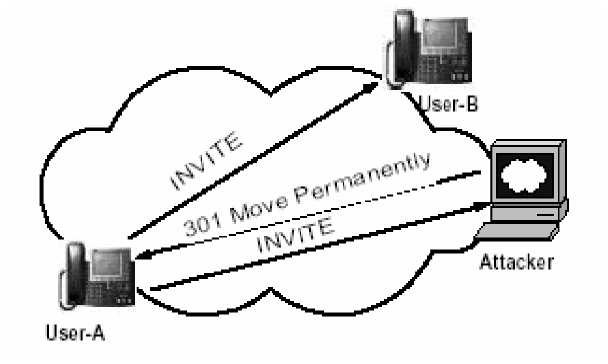


Figure 2. Impersonating Server

2.5 Chosen Plaintext Attack

A chosen plaintext attack is an attack where the cryptanalyst is able to submit his own plaintext, feed it into the cipher, and analyze the resulting ciphertext [17]. The purpose of this type of attack is to find out how a plain message is encrypted.

In SIP, a malicious server can choose the digest parameters (e.g. nonce value) that the client will use to compute the response value. The ability to choose the nonce makes cryptanlysis much easier [3].

Chosen Plaintext attack is very difficult to succeed in SIP digest authentication as SIP uses MD5 algorithm that has been proven to be strong enough for the today's known attacks.

Nevertheless, SIP defines a protection algorithm to prevent this type of attack. The algorithm is that the user agent client (UAC) generates a "cnonce" value to make a stronger checksum. More description on cnonce is in section 3.3.2.

The threats, illustrated in section 2, demonstrate the need for security measures that support authentication between SIP UA and servers. In the next section, we will present how the current SIP authentication mechanisms prevent these threats. These mechanisms, based on the latest SIP standard RFC3261 [1] and the HTTP authentication standard RFC2617 [3], have been widely complied with and adopted by the SIP-enabled products.

3 SIP Authentication

3.1 Overview

During a call involving SIP user-agent and server, an attacker could masquerade as a user, forging the real identity of the sender. Authentication provides a mechanism to verify that a request sender and/or receiver are legitimate.

In a SIP-based network, the authentication can take place between the user agent and the server (proxy, registrar, and user agent server), where the server requires a user agent to authenticate itself before processing the request. Similarly, a user agent can request authentication of a server (known as Mutual Authentication). In general, authentication is needed in the following cases:

- Registration: Malicious users need to be prevented from registering for parties that have not authorized to do so.
- Session setup: During session setup triggered by an INVITE method, each participant desires to know the true identity of the other party.
- Session modification: Even if the session participants do not know or care who the other participant is, they do care that an unauthorized third party does not modify their session in progress.
- Terminating sessions: Terminating a session, via BYE.

In a SIP-based system, authentication measures can be enabled at different layers, including application layer, transport layer, and network layer (shown in Figure 3). The typical protocol used at network layer is IPSec [11]. The protocol used at transport layer is Transport Layer Security (TLS) [4]. Both IPSec and TLS encrypt the messages in transmission. It is impossible to apply a single mechanism to solve all SIP security issues. Each mechanism, such as digest authentication, TLS and IPSec, has different strength and application scope. It is likely that a variety of multi-layer based mechanisms will be deployed, depending on the network infrastructure, restrictions on the computational power of end systems, security expectations, attack models and bandwidth constraints [13].

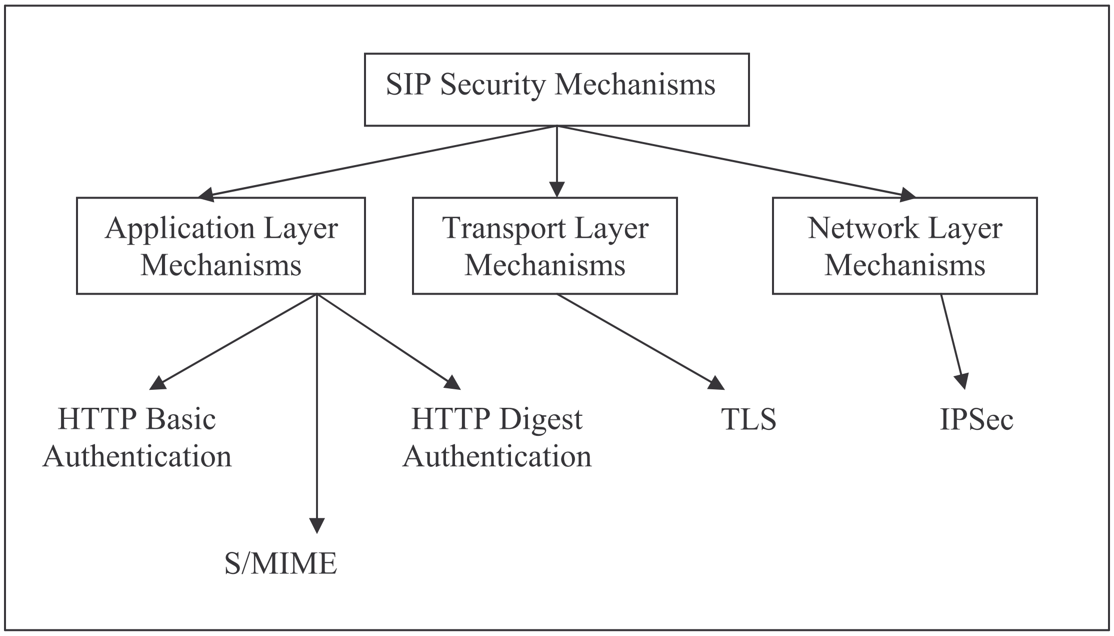
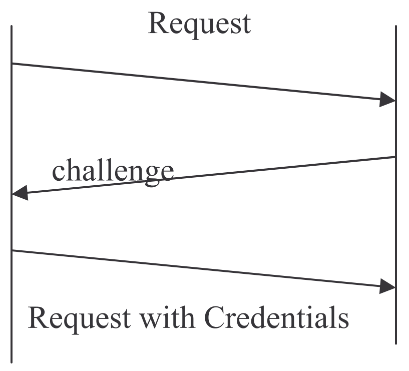

SIP Security Mechanisms
Application Layer Mechanisms
Transport Layer Mechanisms
Network Layer Mechanisms
HTTP Basic Authentication
HTTP Digest Authentication
TLS
IPSec
S/MIME

Figure 3. SIP Security Architecture

Without cryptographic security services from transport and network layer, SIP, as an application level protocol, only provide Digest Authentication service as per the latest SIP standard RFC3261 [1]. Using this mechanism the UAC can identify itself to a UAS or registrar server, or to a next-hop proxy server. Digest authentication also supports the mutual authentication that the server authenticates itself to the UAC. 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. Since our study is focused on SIP itself, the network and transport layer security mechanisms are out of the scope of this document.

3.2 Digest Authentication

SIP provides a challenge-based mechanism for authentication that is based on authentication in HTTP [3]. This scheme is known as Digest authentication due to the use of an MD5 hashing function on the username/password combination.



By digest authentication, when a client tries to establish a connection with UAS, registrar, or redirect server, the server sends the 401 Unauthorized response to challenge the identity of a UAC; when the client initializes a connection with a proxy server, the proxy responds with the 407 Proxy Authentication Required to challenge the UAC. A simple challenge-based authentication is illustrated in Figure 4.

Figure 4. Challenge-based Authentication

3.2.1 Digest Authentication Headers

SIP uses headers for authentication. The WWW-Authenticate header is used in 401 response message sent by the server. In response to the 401 challenge, the UA should include a Authorization header containing the credentials in the next request. Similarly, the Proxy-Authenticate header is used in 407 response message sent by the proxy; and the UA should include a Proxy-Authorization header in the next request.

An example of the WWW-Authenticate header field in a 401 challenge is below. The 407 challenge has the similar parameters.

```
WWW-Authenticate: Digest
      realm="biloxi.com",
      qop="auth,auth-int",
      nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
      opaque="5ccc069c403ebaf9f0171e9517f40e41",
      algorithm=MD5
```

The brief description of each parameter is given in Table 3. Some of headers are optional.

Parameter	Description
Digest	Indicator of authentication scheme
realm	Associated protection domain
qop	Specifying "Quality of Protection" that the server supports. Its value can be either "auth" for authentication, or "auth-int" for authentication and integrity. (Optional)

nonce	Unique string specified by the server.
opaque	String specified by the server for the client to return in the subsequent requests. (optional)
algorithm	The algorithm used for checksum calculation. The default value is MD5 (optional)
stale	A flag indicating if the nonce value in the previous request is stale (optional)

Table 3: Parameters in WWW-Authenticate Header

After the client receives the 401 or 407 challenge from the server, it re-submits a request with the credentials by including an Authorization (in response to 401) or Proxy-Authorization (in response to 407) header field with the request.

An example of the Authorization header field is below. The Proxy-Authorization header has the similar parameters.

```
    Authorization: Digest username="bob",
        realm="biloxi.com",
        nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
        qop=auth,
        nc=00000001,
        cnonce="0a4f113b",
        response="6629fae49393a05397450978507c4ef1",
        opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

The parameter realm, nonce, qop, algorithm, and opaque are identical to those in WWW-Authentication header. The brief description of other parameters is given in Table 4.

Parameter	**Description**
username	The user's name in the specified realm
nc	Count of the number of requests that the client has sent with the nonce value
cnonce	If a qop value is specified, the client must specify cnonce value
response	The string of 32 hex digits calculated checksum

Table 4: Parameters in Authorization Header

3.2.2. Digest Calculation

The method of calculating the Request-Digest is as follows, if qop is present and the default algorithm MD5 is used.

```
request-digest  = <"> < KD ( H(A1),  unq(nonce-value)
                        ":" nc-value
                        ":" unq(cnonce-value)
                        ":" unq(qop-value)
                        ":" H(A2)
                    ) <">

    where
            A1 = unq(username-value) ":" unq(realm-value) ":" passwd
            A2 = Method ":" digest-uri-value
```

The definition of KD, H, and unq are given below:

- KD (secret, data) denotes the string obtained by applying digest algorithm to the data “data” with secret “secret”.

- H(data) denotes the string obtained by applying the checksum algorithm to the data “data”.

- unq(X) denotes the value of the quoted-string X without the surrounding quotes.

3.3. Protection Mechanisms

This section describes the protection measures for client and server authentication against the attacks illustrated in section 2.

3.3.1 Client Authentication

The client authentication scheme is to ensure the client to be legitimate when establishing a connection with a remote server. This scheme can provide the protection against replay attack, request spoofing, and registration hijacking as described in section 2.

SIP defines a “nonce” value to authenticate the client to the server. The “nonce” value is implementation dependent. A recommended digest implementation should generate the “nonce” value with, at least, a digest of client IP address and a time-stamp. This makes a replay attack difficult. If a replay attacker wants to succeed, he must ensure the IP address is what the server expects; and he only has chance to succeed in the period before the time-stamp expires.

The digest containing time-stamp as described above allows the same nonce value to be used repeatedly in the subsequent transactions as long as the time-stamp is expired. This opens a “tiny” hole to replay attacks if the attack is able to capture and use the nonce value that is not expired yet. Thus, if a SIP system is zero-tolerate to the replay attacks, the server has to generate one-time nonce value that is prohibited for a second use. One way of generating one-time nonce value is to use Authenticate-Info or Proxy-Authenticate-Info containing a nextnonce parameter or directive that will be used by the client in the next request. The detail on nextnonce is described in section 3.3.2.2 and 3.3.2.3.

It is worth to note that the use of one-time nonce value may affect the server performance. The pipelined requests cannot be processed by the server if every response includes a nextnonce directive that requires the client to use in the next request. Thus, the implementation should consider the tradeoffs between performance and security. In practice, an old nonce value is often allowed to be used repeated in a limited time period. Another workaround is to use the nonce-count (nc) that can retain most of the security advantages of a new server nonce without affecting pipelined requests.

3.3.2 Server Authentication

On the other side, to ensure the server to have a legitimate identity, the server can authenticate itself to a client. This scheme provides protection against Server impersonating and Chosen Plaintext Attacks as illustrated in section 2.

3.3.2.1 Parameters for Server Authentication

On receipt of 401 or 407 challenge, the UAC is expected to resubmit the request and includes Authorization or Proxy-Authorization header in the request. In the Authorization or Proxy-Authorization header, some of parameters, as described below, are used to provide server authentication.

Note that the use of the following is optional and it is only used when mutual authentication is required. That is, if a qop directive is sent, cnonce and nonce-count value must be specified in Authorization or Proxy-Authorization response; if a qop directive is not sent, cnonce and nonce-count must not be specified.

- cnonce

  SIP defines an optional “cnonce” parameter whose value is generated and stored by the client and sent to the server; the server should include this nonce value in the response header. This approach allows the client to vary the input to the hash rather than chosen by the server. See the example in section 3.3.3 for the use of cnonce.

- nonce-count

nonce-count (nc) is to enable the server to detect message replay attacks by maintaining its own copy of this count. If the same nc value appears twice, then the request is a replay. See the example in section 3.3.3 for the use of cnonce.

3.3.2.2. Authenticate-Info Header

Authenticate-Info header is used by the server to send the client some information regarding the successful authentication in the response. If a request is successfully authenticated by using digest in the Authorization header field, the UAS or registrar server can include this header in a 2xx response.

The most important parameter in this header is nextnonce. The value of the nextnonce is the nonce the server provides for the client to use for the next authentication response. The server sends the Authentication-Info header with a nextnonce as a means of implementing one-time nonce value to prevent replay attacks. An example of the header field is shown as below:

```
Authentication-Info: nextnonce="6324f23987e95131a5fb210812c"
```

3.3.2.3. Proxy-Authenticate-Info Header

The purpose of Proxy-Authenticate-Info header is similar to Authenticate-Info header with one difference: Proxy-Authenticate-Info header is used by the proxy server while Authenticate-Info header is used by the UAS or registrar server. An example of the header field is shown as below:

```
Proxy-Authentication-Info: nextnonce="6324f23987e95131a5fb210812c"
```

3.3.3. An Example of Complete Protection Scheme

This section shows a mutual authentication example shown in Figure 5 that combines all headers and directives illustrated in section 3.3.1 and 3.3.2. Note that the authentication scheme we have seen so far is only applied between UAC and its outbound proxy (proxy1 in the diagram).

In this example, UAC (Alice's computer) initializes a call to UAS (Bob's computer) via two proxies Proxy1 (atlanta.com) and Proxy2 (biloxi.com). The initial INVITE (M1) does not contain the credentials Proxy 1 requires, so a 407 Proxy Authorization response (M2) is sent containing the challenge information. A new INVITE (M4) is then sent containing the correct credentials for the client authentication and the cnonce and nc values for the server authentication. After proxy1 authenticates the UAC each other, the UAS sends 200 OK response all the way back to the UAC, and in the final hop (Proxy1→ UAC), proxy1 inserts the Proxy-Authenticate-Info containing the nextnonce to achieve the one-time nonce value goal that eliminates replay attacks. The call terminates when Bob disconnects by initiating a BYE message.

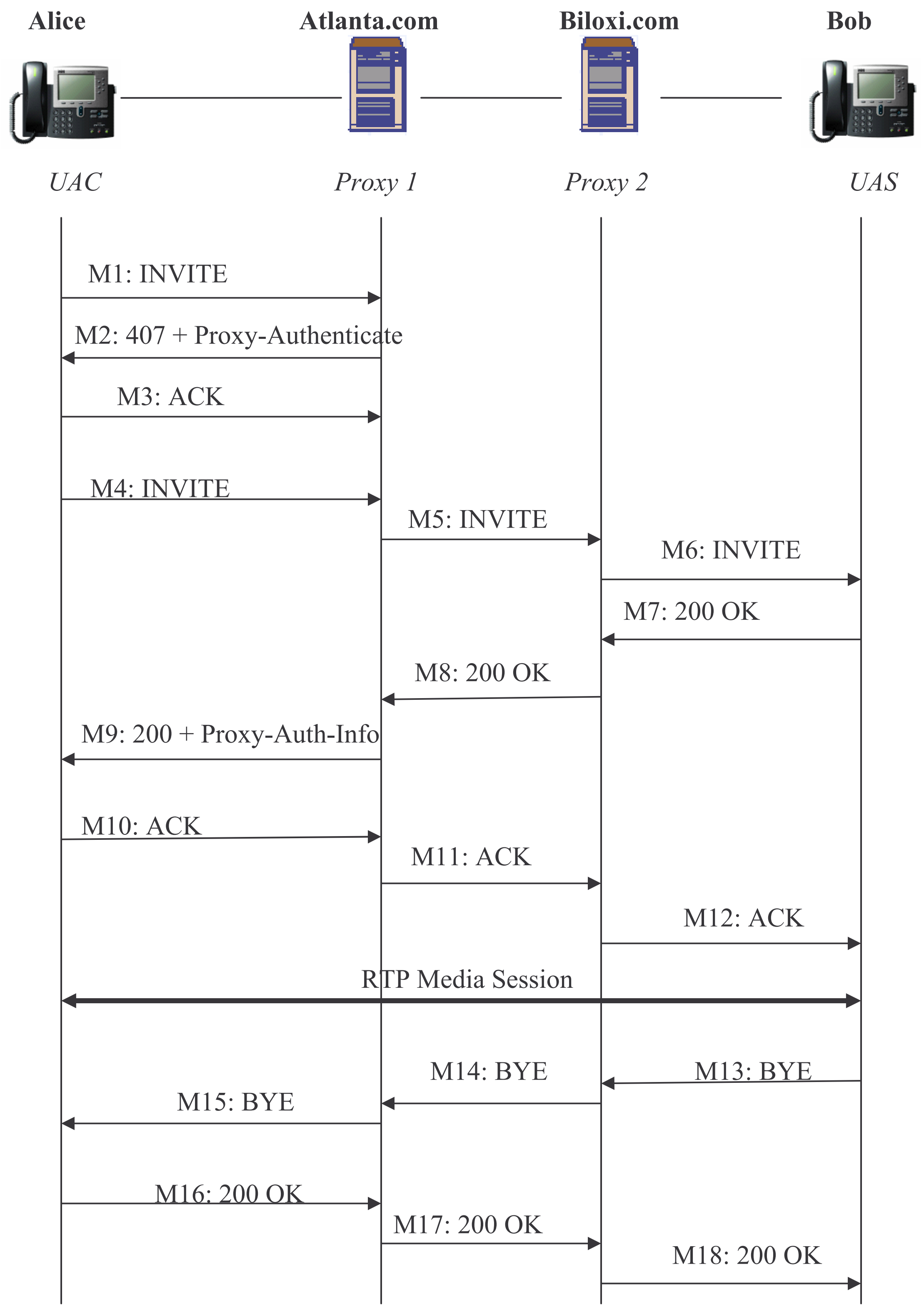


Figure 5. Example Mutual Authentication Data Flow

Message details:

- M1: INVITE message from Alice to proxy1. pc33.atlanta.com is the domain name of UAC.

  ```
  INVITE sip:bob@biloxi.com SIP/2.0
  Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
  Max-Forwards: 70
  To: Bob <sip:bob@biloxi.com>
  From: Alice <sip:alice@atlanta.com>;tag=1928301774
  Call-ID: a84b4c76e66710
  CSeq: 1 INVITE
  Content-Length: 0
  ```

- M2: proxy1 challenges UAC for authentication by including Proxy-Authenticate header field in 407 response. Note that the Proxy-Authenticate header contains the directives such as qop to indicate the subsequent mutual authentication.

  ```
  SIP/2.0 407 Proxy Authentication Required
  Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
  To: Bob <sip:bob@biloxi.com>
  From: Alice <sip:alice@atlanta.com>;tag=1928301774
  Call-ID: a84b4c76e66710
  CSeq: 1 INVITE
  Proxy-Authenticate: Digest
  algorithm=MD5,realm=”Atlanta.com”,nonce=”1d364e211ae3212”,
  qop=”auth”,opaque=”532acb365e232f23ee”
  Content-Length: 0
  ```

- M3: UAC acknowledges the receipt of 407 challenge

  ```
  ACK sip:bob@Biloxi.com SIP/2.0
  Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
  To: Bob <sip:bob@biloxi.com>
  From: Alice <sip:alice@atlanta.com>;tag=1928301774
  Call-ID: a84b4c76e66710
  CSeq: 1 ACK
  Content-Length: 0
  ```

- M4: INVITE message from Alice to proxy1. The UAC includes its credentials in the request. nc and cnonce directives are used to enable mutual authentication: nc allows the server to detect request replays; cnonce is to prevent chosen plaintext attacks.

  ```
  INVITE sip:bob@biloxi.com SIP/2.0
  Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
  Max-Forwards: 70
  ```

```
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 2 INVITE
```
Proxy-Authorization: Digest
username=”alice”,algorithm=MD5,realm=”Atlanta.com”,
nonce=”1d364e211ae3212”,qop=”auth”,opaque=”532acb365e232f23ee”,
response=”839ea122bc3576792”,nc=00000001,cnonce=”3a39aec”
```
Content-Length: 0
```

- M5: the UAC is authenticated successfully, proxy1 forwards the INVITE request to proxy2.

- M6: proxy2 forward the request to UAS.

- M7: Bob answers the call, triggering a 200 OK response back to the UAC. The first stop is proxy2. (server10.biloxi.com is the domain name of proxy2; server1.atlanta.com is the domain name of proxy1)

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server10.biloxi.com
Via: SIP/2.0/UDP server1.atlanta.com
Via: SIP/2.0/UDP pc33.atlanta.com
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 2 INVITE
Content-Length: 0
```

- M8: proxy2 forwards 200 OK to proxy1.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server1.atlanta.com
Via: SIP/2.0/UDP pc33.atlanta.com
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 2 INVITE
Content-Length: 0
```

- M9: proxy1 forwards 200 OK to the UAC and also inserts a Proxy-Authentication-Info header in the 200 OK response.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pc33.atlanta.com
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
```

```
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 2 INVITE
```
Proxy-Authentication-Info: nextnonce="6324f23987e95131a5fb210812c"
```
Content-Length: 0
```

- M10: ACK UAC→ proxy1

```
ACK sip:bob@192.0.2.4 SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 2 ACK
Content-Length: 0
```

- M11: ACK proxy1→ proxy2

```
ACK sip:bob@192.0.2.4 SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com
Via: SIP/2.0/UDP server1.atlanta.com
Via: SIP/2.0/UDP server10.biloxi.com
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 2 ACK
Content-Length: 0
```

- M12: ACK proxy2→ UAS

```
ACK sip:bob@192.0.2.4 SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com
Via: SIP/2.0/UDP server1.atlanta.com
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 2 ACK
Content-Length: 0
```

[The media session between Alice and Bob is now established. At one point, Bob hangs up first].

- M13: BYE message is sent from UAS → proxy2. It is possible for proxy2 to authenticate the UAS in order to prevent the tear-down attacks (request spoofing). The same header fields can be used as in M2 and M4. The use of such authentication is not shown in this example due to space restriction.

  ```
  BYE sip:alice@pc33.atlanta.com SIP/2.0
  Via: SIP/2.0/UDP 192.0.2.4
  Via: SIP/2.0/UDP server10.biloxi.com
  Via: SIP/2.0/UDP server1.atlanta.com
  Max-Forwards: 70
  From: Bob <sip:bob@biloxi.com>;tag=a6c85cf
  To: Alice <sip:alice@atlanta.com>;tag=1928301774
  Call-ID: a84b4c76e66710
  CSeq: 2 BYE
  Content-Length: 0
  ```

- M14: BYE proxy2 → proxy1

  ```
  BYE sip:alice@pc33.atlanta.com SIP/2.0
  Via: SIP/2.0/UDP server10.biloxi.com
  Via: SIP/2.0/UDP server1.atlanta.com
  Max-Forwards: 70
  From: Bob <sip:bob@biloxi.com>;tag=a6c85cf
  To: Alice <sip:alice@atlanta.com>;tag=1928301774
  Call-ID: a84b4c76e66710
  CSeq: 2 BYE
  Content-Length: 0
  ```

- M15: BYE proxy1 → UAC

  ```
  BYE sip:alice@pc33.atlanta.com SIP/2.0
  Via: SIP/2.0/UDP server1.atlanta.com
  Max-Forwards: 70
  From: Bob <sip:bob@biloxi.com>;tag=a6c85cf
  To: Alice <sip:alice@atlanta.com>;tag=1928301774
  Call-ID: a84b4c76e66710
  CSeq: 2 BYE
  Content-Length: 0
  ```

- M16: 200 OK UAC → proxy1

  ```
  SIP/2.0 200 OK
  Via: SIP/2.0/UDP pc33.atlanta.com
  Via: SIP/2.0/UDP server1.atlanta.com
  Via: SIP/2.0/UDP server10.biloxi.com
  To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
  ```

```
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 2 INVITE
Content-Length: 0
```

- M17: 200 OK proxy1 → proxy2

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server1.atlanta.com
Via: SIP/2.0/UDP server10.biloxi.com
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 2 INVITE
Content-Length: 0
```

- M18: 200 OK proxy2 → UAS

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server10.biloxi.com
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 2 INVITE
Content-Length: 0
```

4. Proposed Extension to Digest Authentication

The digest authentication scheme described in section 3 can only be deployed on the link associated with UAC, including UAC to UAS, UAC to registrar server, UAC to proxy server, and UAC to redirect server. This scheme, however, does not provide authentication service on the last hop in a SIP call control routing. This last hop is the link between proxy server and UAS. It is worth to enable proxy-to-UAS authentication as the lack of such authentication is definitely a defect of SIP security if no lower layer security protocols such as TLS and IPSec are used. In the example illustrated in section 3.3.3, to have a complete authentication protection coverage in a call routing, the proxy2 should authenticate itself to the UAS based on the similar challenge-response method when forwarding the message M6.

The current SIP authentication scheme does not provide the headers and parameters for proxy-to-UAS authentication. This section attempts to fix the proxy-to-UAS authentication problem by introducing an extension of SIP digest authentication based on [6]. The extension defines the following new headers and message codes to enable proxy-to-UAS authentication.

4.1. 492 Proxies Unauthorized

If a UAS receives an initial request from the inbound proxy, the UAS will respond with a "492 Proxy Unauthorized" to challenge the proxy. This 492 challenge includes a new header "UAS-Authenticate" that is described in 4.2.1.

4.2. New Headers Specification

4.2.1 UAS-Authenticate Response Header

```
UAS-Authenticate = "UAS-Authenticate" HCOLON 1#uas-challenge
uas-challenge = scheme target-param challenge
target-param = target-realm-param | target-route-param
target-realm-param = "target" EQUAL target
target = host
target-route-param = "route" EQUAL target-route
target-route = Request-URI
```

target
  The target is a hostname or IP address of the targeted proxy domain.

target-route
  It is the uri for a targeted entity that sends a request with this uri as the Request-URI.

The use of this header is described in section 4.3.

4.2.2 UAS-Authorization Request Header

This header is sent together with the re-submitted request by proxy upon receiving the UAS-Authenticate header in the 492 challenge. The syntax is:

```
UAS-Authorization = "UAS-Authorization" HCOLON 1#uas-credentials
uas-credentials = scheme target-ids credentials
target-ids = target-param responder-param
responder-param = "responder" EQUAL responder
responder = sent-by
```

responder
  The responder is hostname or IP address that matches the value inserted in the Via by the proxy.

The use of this header is described in section 4.3.

4.2.3 UAS-Authenticate-Info

The UAS uses this header to authenticate itself to the proxy in the response.

```
UASAuthenticationInfo = "UAS-Authentication-Info" HCOLON
                              1#uas-reverse-credentials
uas-reverse-credentials = scheme target-param info-credentials
```

The above header fields are similar to those defined in Authenticate-Info header in section 2.

4.3 Proxy-to-UAS Authentication Operation

In a call control that originates from a UAC, goes through one or more intermediary proxies, and ends at a UAS, the UAS can authenticate the proxies by sending the proxies a 492 challenge that includes UAS-Authenticate header.

When the proxy receives a 492 challenge, it checks the UAS-Authenticate header to see if the target parameter or route parameter matches. And then the proxy re-submits the UAS the request that contains an Authorization header with credentials for the matched UAS-Authenticate headers.

If the proxy is successfully authenticated, the UAS can perform mutual authentication by using the Proxy-Authentication-Info header in the response (e.g. 200 OK response).

The data flow diagram in Figure 6 shows how the UAS-Authenticate header is used to provide proxy to UAS authentication.

- UAC sends a INVITE message to UAS via proxy
- Upon receiving the request, the UAS returns a 492 Proxy Unauthorized to challenge the proxy. As a complete process, this challenge is propagated back to the UAC.
- The UAC copies the UAS-Authenticate headers into the resubmitted request.
- When the request with the UAS-Authenticate headers arrive at the proxy it adds UAS-Authorization headers for all the challenges targeted at it.
- When the UAS receives request it ensures it has received all the UAS-Authorization headers it was expecting.
- The UAS then populates UAS-Authentication-Info headers for all the proxies it wishes to mutually authenticate with. Proxies can check for UAS-Authentication-Info headers in the response.

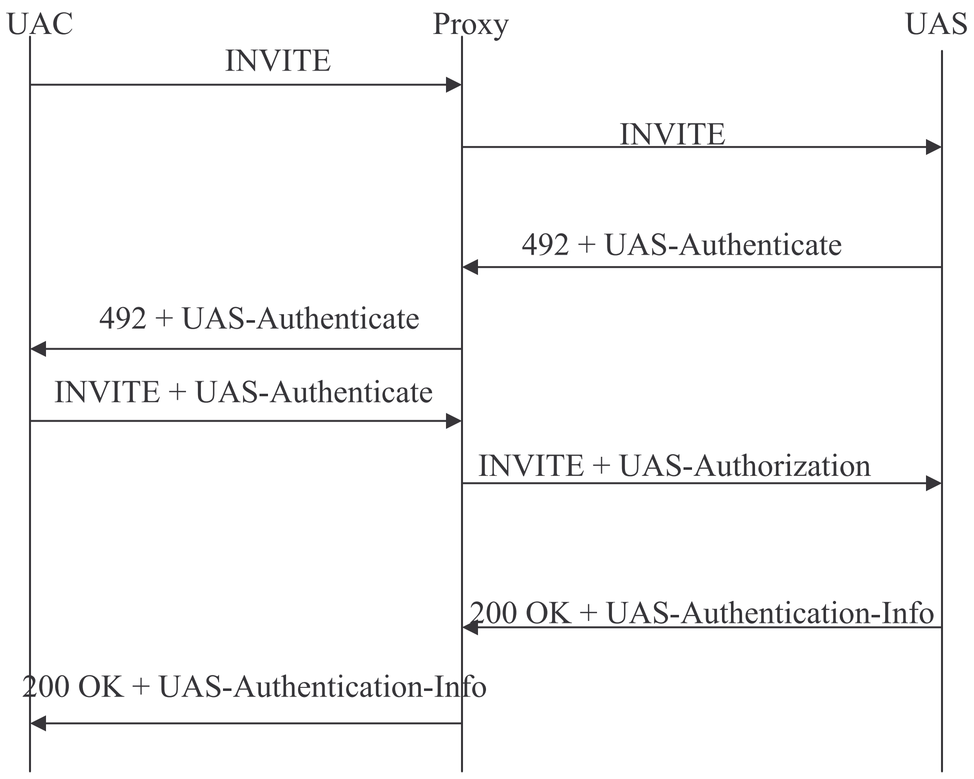

UAC
Proxy
UAS
INVITE
INVITE
492 + UAS-Authenticate
492 + UAS-Authenticate
INVITE + UAS-Authenticate
INVITE + UAS-Authorization
200 OK + UAS-Authentication-Info
200 OK + UAS-Authentication-Info

Figure 6. Proxy to UAS Mutual Authentication

5. Limitations and Future Work

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. These limitations can also be pointers for the future study.

First, the Digest authentication claims that if the value of qop parameter is specified as "auth-int", the integrity protection mechanism will be enabled. However, such mechanism does not work well for SIP since it offers protection only for some SIP parameters.

Second, Digest requires that a preexisting secure association can be used in SIP servers where the user is pre-configured. Thus, digest authentication may not be suitable to the non-secure association such as proxy-to-proxy across different domains on the Internet. The authentication in such case has to rely on TLS or IPSec.

Third, Digest authentication does not provide encryption mechanism due to its nature. Therefore, if SIP messages need privacy protection, Digest authentication should not be considered.

Lastly, like the proposed Digest authentication scheme, any extensions may introduce more and more headers or parameters, this will slow down performance of the SIP. Thus, the tradeoffs between performance and security functionality must be taken into consideration.

Reference

[1] Rosenberg, J, Schulzrinne, H, Camarillo, G, et al. *SIP: Session Initiation Protocol, RFC3261,* June 2002.

[2] Handley, M., Schulzrinne, H., Schooler, E. and J. Rosenberg. *SIP: Session Initiation Protocol, RFC 2543*, March 1999.

[3] Franks, et al. *HTTP Authentication: Basic and Digest Access Authentication*. RFC 2617, June 1999.

[4] Dierks, T., Allen, C. *The TLS Protocol, Version 1.0* RFC 2246, January 1999.

[5] Michael Thomas, SIP Security Framework, draft-thomas-sip-sec-framework-00.txt. July 12, 2001

[6] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, D. Gurle. RFC 3428 - Session Initiation Protocol (SIP) Extension for Instant Messaging, December 2002.

[7] H. Schulzrinne, AMinimalist Security Framework for SIP, draft-schulzrinne-sip-security-00.txt. November 18, 2001.

[8] J. Undery, S. Sen, V. Torvinen. *SIP Digest Authentication: Extensions to HTTP Digest Authentication*, draft-undery-sip-auth-00.txt. January 2002.

[9] Bellovin, S., "Report of the IAB Security Architecture Workshop",
RFC 2316, April 1998.

[10] Dusse, S, et al. "S/MIME Version 2 Message Specification", RFC 2311,
March 1998.

[11] S. Kent, R. Atkinson. Security Architecture for the Internet Protocol, RFC2401. November 1998.

[12] Cisco whitepaper: Security in SIP-Based Networks.
http://www.cisco.com/warp/public/cc/techno/ tyvdve/sip/prodlit/sipsc_wp.pdf. Accessed in Nov. 2003.

[13] D. Comer, Computer and Networks with Internet Applications, Pearson Prentice Hall, 2003.

[14] I. Dalgic, H. Fang. Comparison of H.323 and SIP for IP Telephony Signaling, http://www.nostech.co.kr/reference/data/voip/Comparison%20of%20H.323%20and%20SIP.pdf. Accessed in December 2003.

[15] Fielding, R., Gettys, J., Mogul, J., Frysyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[16] W. Stallings. Cryptography and Network Security: Principles and Pratices, 2nd edition. Prentice-Hall, June 1998.

[17] http://www.linuxsecurity.com/docs/Hack-FAQ/cryptology-04.shtml. Accessed in December 2003.

[18] Rivest, R., “The MD5 Message-Digest Algorithm”, RFC 1321, April 1992.