Žilinská univerzita v Žiline, Fakulta riadenia a informatiky

Bezpečná IP telefónia

Vladislav Brienik   Marek Kapičák

Obsah

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. There are two main standards for signalling. The first one is H.323 protocol suite proposed by the international telecommunication union. The second one is the session initiation protocol proposed by the internet engineering task force. Both protocols possess its own strengths in different aspects, but they provide similar mechanisms for call establisment, 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 instatn 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 os 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 request. A SIP-based system s 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 place 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 message

SIP is a text-based protocol that is based on the http protocol. 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. Next table 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 users location	YES

1.2.2 Response Message

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. Next table 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: SIP status code

1.3 SIP functionality

End users are identified by SIP URLs. A SIP URL of 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, from 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.

2. Security Threats an 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.

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) 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. 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 capturess all the messages sent in the session. Then, C tries to re-send the message 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 user to the attacker’s device.

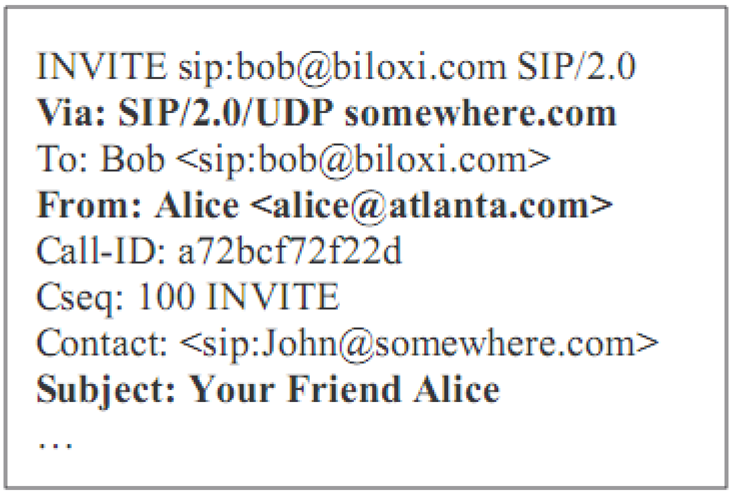
2.3 Request spoofing

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
...
```

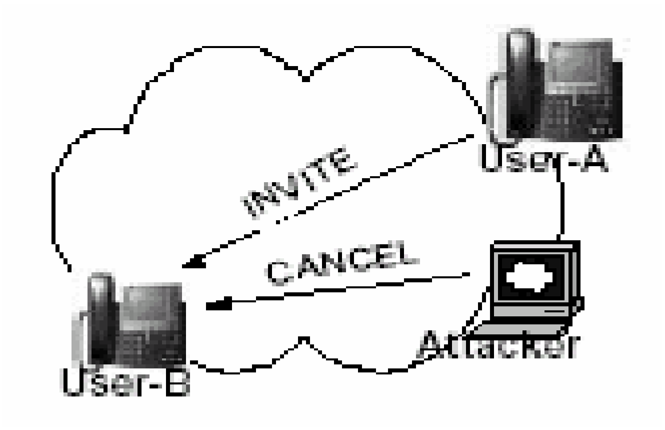
In this example, firstly, John is the attacker who wants Bob to believe that the message is from Alice, therefore he puts Alice's address in the From header field instead of his. Secondly, since John's intension is to establish a dialog with Bob with a fake identity, which means that John wants Bob to send the response back to him rather than Alice, John has to specify his own IP address or domain name in Via headeer 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

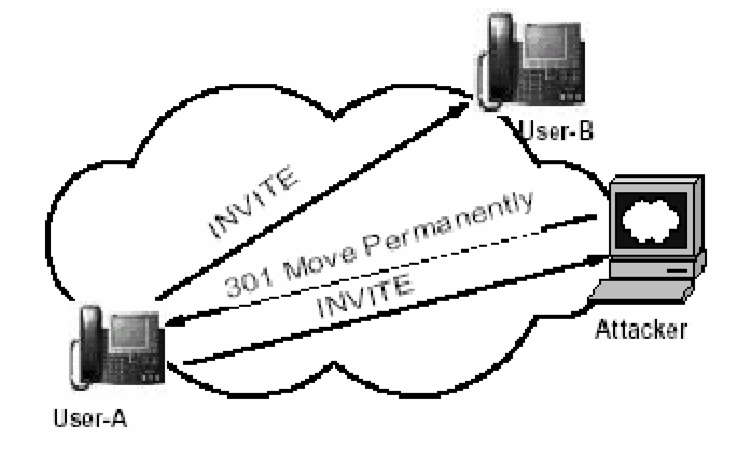
The purpose of this attack is to stop a legitimate INVITE request by sending a forged CANCEL message to the legitimate callee (pic. 1), so that the legitimate INVITE message will be treated as an invalid request.



2.4 Impersonating server

The threats describe 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 (pic. 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.



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. 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.

Chosen Plaintext attack is very difficult to succed 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 3.2.2.

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 RFC 3261 and the HTTP authentication standard RFC 2617, have been widely complied with and adopted by the SIP-enabled products.

3.SIP Authentication

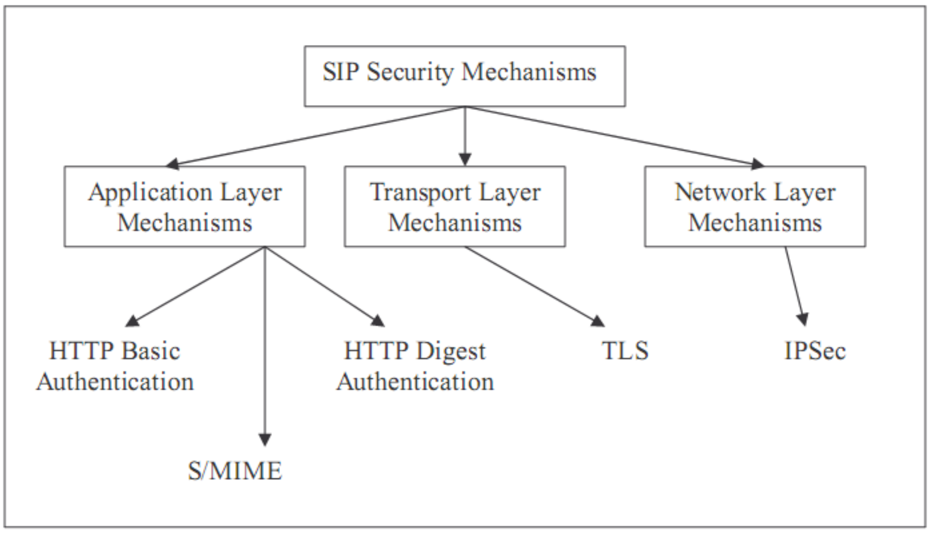
3.1 Overview

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 picture 3). The typical protocol used at network layer is IPSec. The protocol used at transport layer is Transport Layer Security (TLS). 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.


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

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 RFC 3261. 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. 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 send 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.

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 proxxy 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
```

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 (Optionl)
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)

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 next table.

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

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 obrained 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-tolerated 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-Authenticacte-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 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

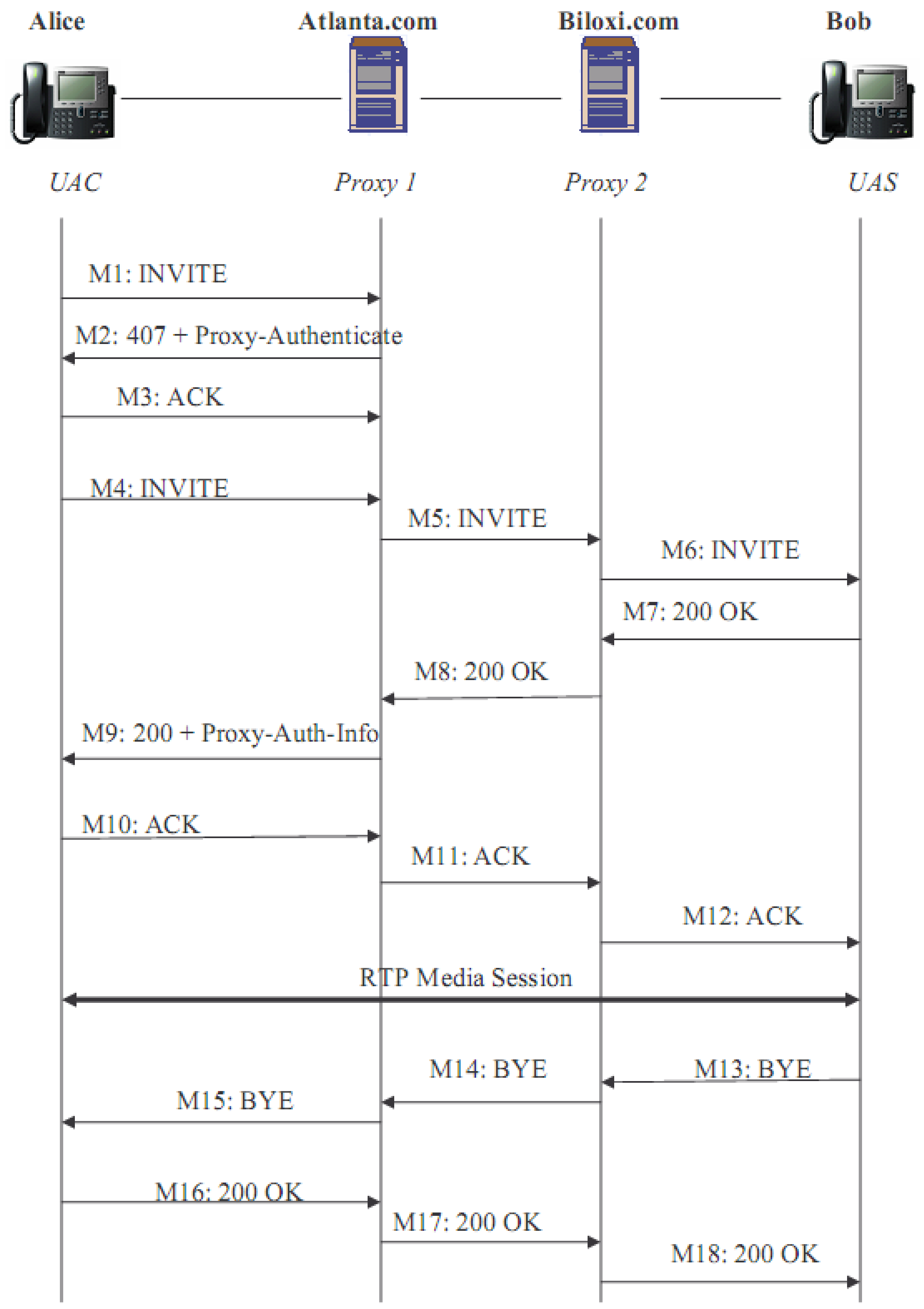
The purpose of Proxy-Authenticate-Info header is similar to Authenticate-Info header with one difference: Proxy-Authenticate-Info header is used by the proxy server while Authenticate-Info header is used by the UAS or registrar server. An example of the header field is shown as below:

Proxy-Authentication-Info: nextnonce=“6324f23987e95131a5fb210812c“

3.3.3 An example of complete protection scheme

This section shows a mutual authentication example shown in next picture 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 (proxy 1 in the diagram).

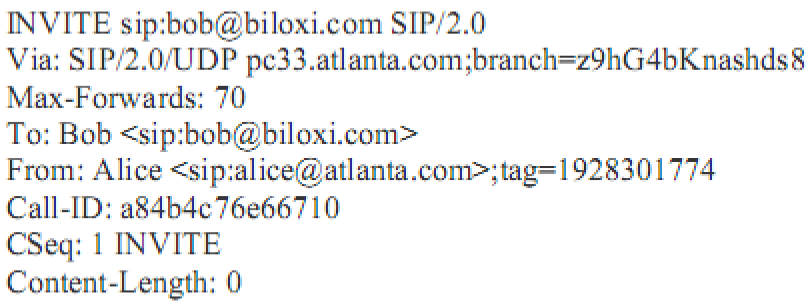
In this example, UAC (Alice’s computer) initializes a call to UAS (Bob’s computer) via two proxies Proxy1 (atlanta.com) and Proxy2 (biloxi.com). The initial INVITE (M1) does not contain the credentials Proxy 1 requires, so a 407 Proxy Authorization response (M2) is sent containing the challenge information. 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 proxy 1 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.



Example Mutual Authentication Data Flow

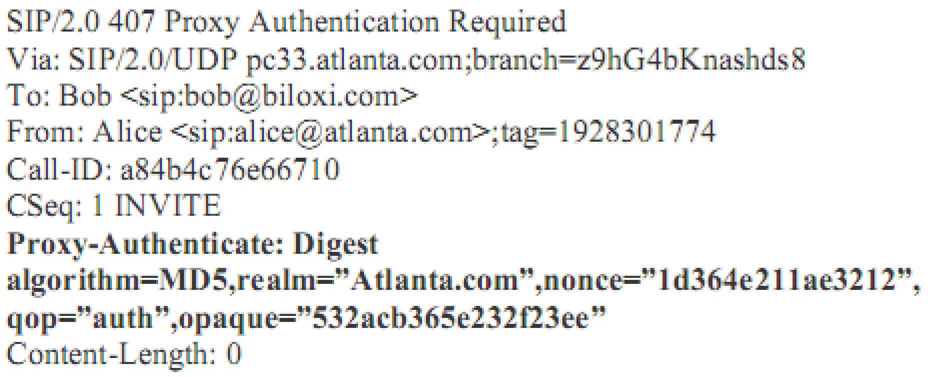
Message details:

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



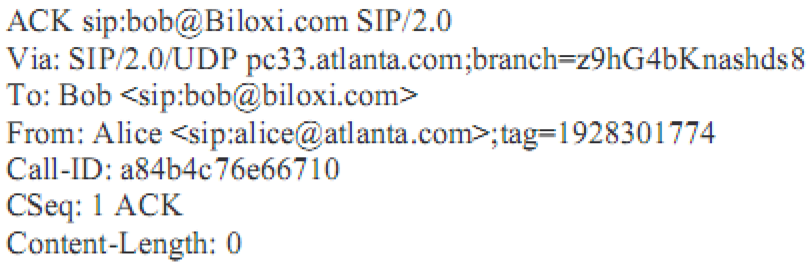
```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 1 INVITE
Content-Length: 0
```

- 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

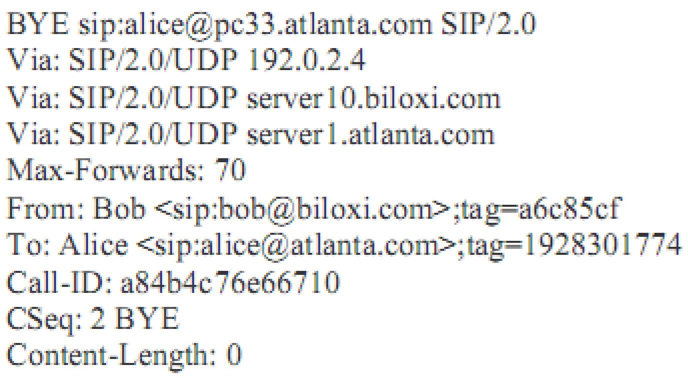
```
ACK sip:bob@192.0.2.4 SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com
Via: SIP/2.0/UDP server1.atlanta.com
Via: SIP/2.0/UDP server10.biloxi.com
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 2 ACK
Content-Length: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

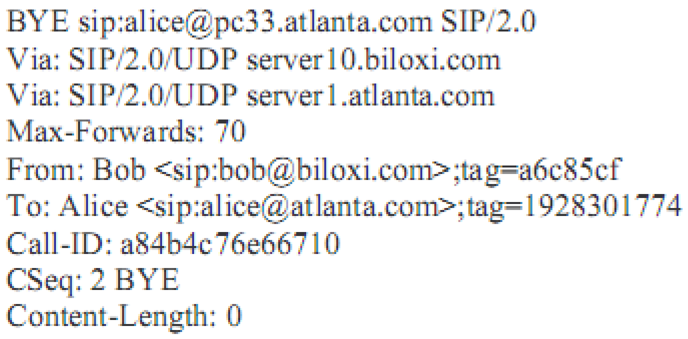
The media session between Alice and Bob is now established. At one point, Bob hangs up first.

- 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.



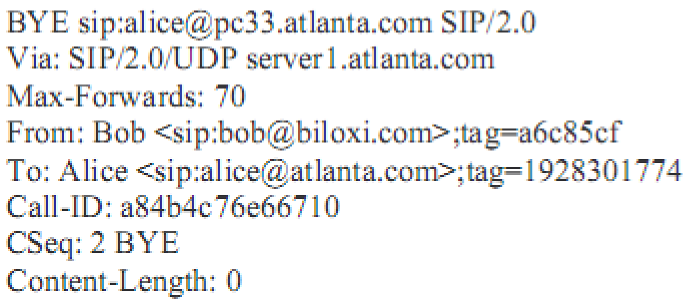
```
BYE sip:alice@pc33.atlanta.com SIP/2.0
Via: SIP/2.0/UDP 192.0.2.4
Via: SIP/2.0/UDP server10.biloxi.com
Via: SIP/2.0/UDP server1.atlanta.com
Max-Forwards: 70
From: Bob <sip:bob@biloxi.com>;tag=a6c85cf
To: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 2 BYE
Content-Length: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

Itis 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.

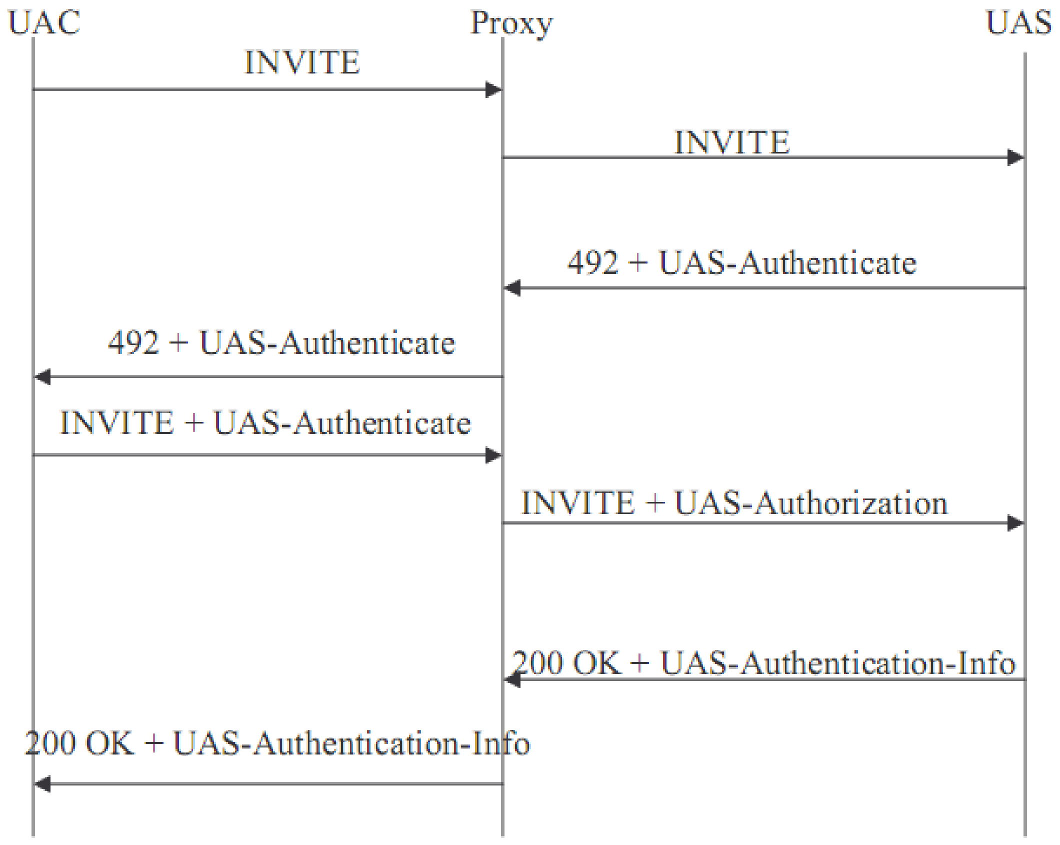
When the proxy receives a 492 challenge, it checks the UAS-authenticate header to see if the target parameter or route parameter matches. And then the proxy re-submits the UAS the request that contains an Authorization header with credentials for the matched UAS-Authenticate headers.

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 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 challnge 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

Proxy to UAS mutual authentication

5. S/MIME

SIP messages carry MIME bodies and the MIME standard includes mechanisms for securing MIME contents to ensure both integrity and confidentiality (including the 'multipart/signed' and 'application/pkcs7-mime' MIME types, see RFC 1847^{prop} [22], RFC 2630(-> 3370^{prop} | 3369(-> 3852^{prop})) [23] and RFC 2633(-> 3851^{prop}) [24]). Implementers should note, however, that there may be rare network intermediaries (not typical proxy servers) that rely on viewing or modifying the bodies of SIP messages (especially SDP), and that secure MIME may prevent these sorts of intermediaries from functioning.

This applies particularly to certain types of firewalls.

5.1 S/MIME Certificates

The certificates that are used to identify an end-user for the purposes of S/MIME differ from those used by servers in one important respect - rather than asserting that the identity of the holder corresponds to a particular hostname, these certificates assert that the holder is identified by an end-user address. This address is composed of the concatenation of the "userinfo" "@" and "domainname" portions of a SIP or SIPS URI (in other words, an email address of the form "bob@biloxi.com"), most commonly corresponding to a user's address-of-record.

These certificates are also associated with keys that are used to sign or encrypt bodies of SIP messages. Bodies are signed with the private key of the sender (who may include their public key with the message as appropriate), but bodies are encrypted with the public key of the intended recipient. Obviously, senders must have foreknowledge of the public key of recipients in order to encrypt message bodies. Public keys can be stored within a UA on a virtual keyring. Each user agent that supports S/MIME MUST contain a keyring specifically for end-users' certificates. This keyring should map between addresses of record and corresponding certificates. Over time, users SHOULD use the same certificate when they populate the originating URI of signaling (the From header field) with the same address-of-record. Any mechanisms depending on the existence of end-user certificates are seriously limited in that there is virtually no consolidated authority today that provides certificates for end-user applications. However, users SHOULD acquire certificates from known public certificate authorities. As an alternative, users MAY create self-signed certificates. The implications of self-signed certificates are explored further in Section 26.4.2. Implementations may also use pre-configured certificates in deployments in which a previous trust relationship exists between all SIP entities. Above and beyond the problem of acquiring an end-user certificate, there are few well-known centralized directories that distribute end-user certificates. However, the holder of a certificate SHOULD publish their certificate in any public directories as appropriate. Similarly, UACs SHOULD support a mechanism for importing (manually or automatically) certificates discovered in public directories corresponding to the target URIs of SIP requests.

5.2 S/MIME Key Exchange

SIP itself can also be used as a means to distribute public keys in the following manner.

Whenever the CMS SignedData message is used in S/MIME for SIP, it MUST contain the certificate bearing the public key necessary to verify the signature.
When a UAC sends a request containing an S/MIME body that initiates a dialog, or sends a non-INVITE request outside the context of a dialog, the UAC SHOULD structure the body as an S/MIME 'multipart/signed' CMS SignedData body. If the desired CMS service is EnvelopedData (and the public key of the target user is known), the UAC SHOULD send the EnvelopedData message encapsulated within a SignedData message.

When a UAS receives a request containing an S/MIME CMS body that includes a certificate, the UAS SHOULD first validate the certificate, if possible, with any available root certificates for certificate authorities. The UAS SHOULD also determine the subject of the certificate (for S/MIME, the SubjectAltName will contain the appropriate identity) and compare this value to the From header field of the request. If the certificate cannot be verified, because it is self-signed, or signed by no known authority, or if it is verifiable but its subject does not correspond to the From header field of request, the UAS MUST notify its user of the status of the certificate (including the subject of the certificate, its signer, and any key fingerprint information) and request explicit permission before proceeding. If the certificate was successfully verified and the subject of the certificate corresponds to the From header field of the SIP request, or if the user (after notification) explicitly authorizes the use of the certificate, the UAS SHOULD add this certificate to a local keyring, indexed by the address-of-record of the holder of the certificate.

When a UAS sends a response containing an S/MIME body that answers the first request in a dialog, or a response to a non-INVITE request outside the context of a dialog, the UAS SHOULD structure the body as an S/MIME 'multipart/signed' CMS SignedData body. If the desired CMS service is EnvelopedData, the UAS SHOULD send the EnvelopedData message encapsulated within a SignedData message.

When a UAC receives a response containing an S/MIME CMS body that includes a certificate, the UAC SHOULD first validate the certificate, if possible, with any appropriate root certificate. The UAC SHOULD also determine the subject of the certificate and compare this value to the To field of the response; although the two may very well be different, and this is not necessarily indicative of a security breach. If the certificate cannot be verified because it is self-signed, or signed by no known authority, the UAC MUST notify its user of the status of the certificate (including the subject of the certificate, its signator, and any key fingerprint information) and request explicit permission before proceeding. If the certificate was successfully verified, and the subject of the certificate corresponds to the To header field in the response, or if the user (after notification) explicitly authorizes the use of the certificate, the UAC SHOULD add this certificate to a local keyring, indexed by the address-of-record of the holder of the certificate. If the UAC had not transmitted its own certificate to the UAS in any previous transaction, it SHOULD use a CMS SignedData body for its next request or response.

On future occasions, when the UA receives requests or responses that contain a From header field corresponding to a value in its keyring, the UA SHOULD compare the certificate offered in these messages with the existing certificate in its keyring. If there is a discrepancy, the UA MUST notify its user of a change of the certificate (preferably in terms that indicate that this is a potential security breach) and acquire the user's permission before continuing to process the signaling. If the user authorizes this certificate, it SHOULD be added to the keyring alongside any previous value(s) for this address-of-record. Note well however, that this key exchange mechanism does not guarantee the secure exchange of keys when self-signed certificates, or certificates signed by an obscure authority, are used - it is vulnerable to well-known attacks. In the opinion of the authors, however, the security it provides is proverbially better than nothing; it is in fact comparable to the widely used SSH application. These limitations are explored in greater detail in Section 26.4.2. If a UA receives an S/MIME body that has been encrypted with a public key unknown to the recipient, it MUST reject the request with a 493 (Undecipherable) response. This response SHOULD contain a valid certificate for the respondent (corresponding, if possible, to any address of record given in the To header field of the rejected request) within a MIME body with a 'certs-only' "smime-type" parameter.

A 493 (Undecipherable) sent without any certificate indicates that the respondent cannot or will not utilize S/MIME encrypted messages, though they may still support S/MIME signatures.

Note that a user agent that receives a request containing an S/MIME body that is not optional with a Content-Disposition header "handling" parameter of "required") MUST reject the request with a 415 Unsupported Media Type response if the MIME type is not understood. A user agent that receives such a response when S/MIME is sent SHOULD notify its user that the remote device does not support S/MIME, and it MAY subsequently resend the request without S/MIME, if appropriate; however, this 415 response may constitute a downgrade attack.

If a user agent sends an S/MIME body in a request, but receives a response that contains a MIME body that is not secured, the UAC SHOULD notify its user that the session could not be secured. However, if a user agent that supports S/MIME receives a request with an unsecured body, it SHOULD NOT respond with a secured body, but if it expects S/MIME from the sender (for example, because the sender's From header field value corresponds to an identity on its keychain), the UAS SHOULD notify its user that the session could not be secured.

A number of conditions that arise in the previous text call for the notification of the user when an anomalous certificate-management event occurs. Users might well ask what they should do under these circumstances. First and foremost, an unexpected change in a certificate, or an absence of security when security is expected, are causes for caution but not necessarily indications that an attack is in progress. Users might abort any connection attempt or refuse a connection request they have received; in telephony parlance, they could hang up and call back. Users may wish to find an alternate means to contact the other party and confirm that their key has legitimately changed. Note that users are sometimes compelled to change their certificates, for example when they suspect that the secrecy of their private key has been compromised. When their private key is no longer private, users must legitimately generate a new key and re-establish trust with any users that held their old key.
Finally, if during the course of a dialog a UA receives a certificate in a CMS SignedData message that does not correspond with the certificates previously exchanged during a dialog, the UA MUST notify its user of the change, preferably in terms that indicate that this is a potential security breach.

5.3 Securing MIME bodies

There are two types of secure MIME bodies that are of interest to SIP: use of these bodies should follow the S/MIME specification [24] with a few variations.

o "multipart/signed" MUST be used only with CMS detached signatures.

This allows backwards compatibility with non-S/MIME- compliant recipients.

o S/MIME bodies SHOULD have a Content-Disposition header field, and the value of the "handling" parameter SHOULD be "required."

o If a UAC has no certificate on its keyring associated with the address-of-record to which it wants to send a request, it cannot send an encrypted "application/pkcs7-mime" MIME message. UACs MAY send an initial request such as an OPTIONS message with a CMS detached signature in order to solicit the certificate of the remote side (the signature SHOULD be over a "message/sip" body of the type described in Section 23.4).

> Note that future standardization work on S/MIME may define
> non-certificate based keys.

o Senders of S/MIME bodies SHOULD use the "SMIMECapabilities" attribute to express their capabilities and preferences for further communications. Note especially that senders MAY use the "preferSignedData" capability to encourage receivers to respond with CMS SignedData messages (for example, when sending an OPTIONS request as described above).

o S/MIME implementations MUST at a minimum support SHA1 as a digital signature algorithm, and 3DES as an encryption algorithm. All other signature and encryption algorithms MAY be supported. Implementations can negotiate support for these algorithms with the "SMIMECapabilities" attribute.

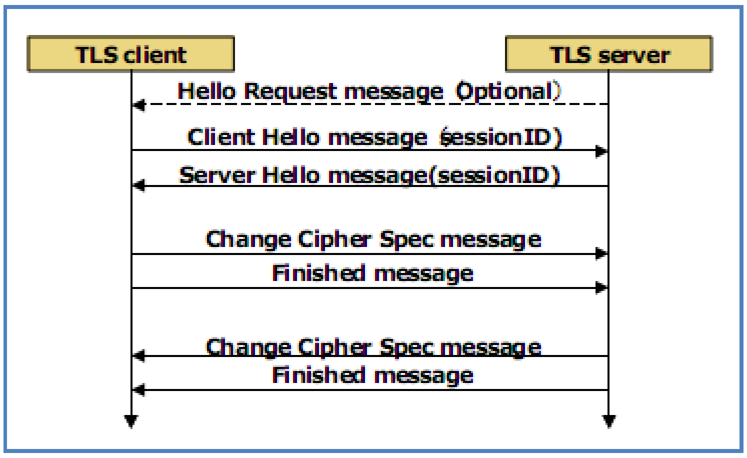
o Each S/MIME body in a SIP message SHOULD be signed with only one certificate. If a UA receives a message with multiple signatures, the outermost signature should be treated as the single certificate for this body. Parallel signatures SHOULD NOT be used.

The following is an example of an encrypted S/MIME SDP body within a SIP message:

```
INVITE 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: 314159 INVITE
Max-Forwards: 70
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
    name=smime.p7m
Content-Disposition: attachment; filename=smime.p7m
  handling=required
```

6 Problems of TLS handshake

TLS performs "handshake process" before starting application data exchange between TLS server and TLS client.

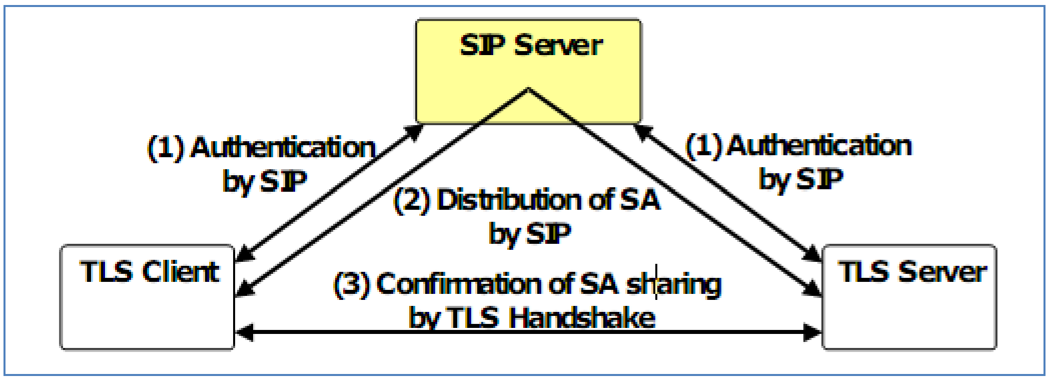

TLS client
TLS server
Hello Request message (Optional)
Client Hello message (sessionID)
Server Hello message(sessionID)
Change Cipher Spec message
Finished message
Change Cipher Spec message
Finished message

Pict: Exchanged messages for resuming session

The problem of the TLS session resume is that the resuming session skips not only the peer authentication, but also both of the authorization of session and the negotiation of security association. However, in many cases, the authorization of the session and the negotiation of security do not depend on peer entity, but depends on certain session. Therefore the authorization and negotiation processes are required to perform by each session.

6.1 Proposed TLS handshake method based on SIP

The basic idea of the proposed TLS handshake is to introduce a SIP server as the entity that performs costly process (peer authentication and security association generation) for TLS session on behalf of TLS client and TLS server to reduce these costs.


SIP Server
(1) Authentication by SIP
(1) Authentication by SIP
(2) Distribution of SA by SIP
TLS Client
TLS Server
(3) Confirmation of SA sharing by TLS Handshake

Pict1: Overview of proposed method

To establish a TLS session between TLS server and TLS client, the proposed method takes three phases: authentication, key distribution and SA confirmation.

At first, authentication phase is performed. SIP server authenticates TLS client and TLS server respectively. If authentication is success, SIP server establishes and keeps SIP sessions with TLS client and TLS server. AND TLS client (TLS server) register security policy to SIP server via the SIP session.

Next, key distribution phase is performed. SIP server generates a security association for TLS session between TLS client and TLS server and distributes it to TLS client and TLS server through the SIP session. This phase is triggered when TLS client tries to communicate with TLS server.

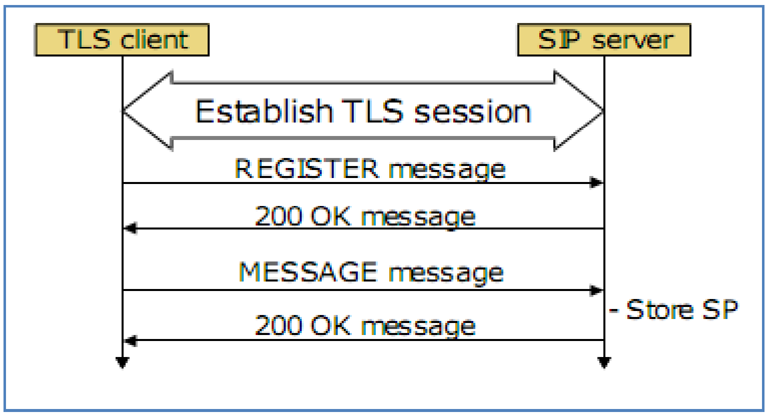
And then, SA confirmation phase is performed. As the resuming session procedure, TLS server and TLS client confirm if they share a security association. This phase follows the key distribution phase immediately. If this phase finishes successfully, TLS server and TLS client starts the application data transmission over the established TLS session.

It notes that the SIP message exchanged in the proposed method will not contain SDP message, but contain XML to transmit security policy and security association.

6.2 Authentication phase

6.2.1 Message sequences and system behaviors

In authentication phase, TLS client (or TLS server) establish a TLS session with SIP server in the normal way at first. And then they exchange some SIP messages via the TLS session.



Pict2: Shows message sequence during authentication phase of TLS client.

1) At first, TLS client establishes a TLS session with SIP server in the normal way. Mutual authentication is performed during TLS session establishment.

2) Then, TLS client sends a REGISTER message to SIP server.

3) SIP server authenticates a sender of REGISTER message (i.e. TLS client), and returns a response message (200 OK) if authentication is success.

4) TLS client authenticates a sender of the response message (i.e. SIP server) when the message is received.

5) And if authentication is success, MESSAGE message with security policy (i.e, a set of available cipher suites for TLS session) are sent to SIP server for key distribution phase.

6) SIP server stores the security policy and returns a response message.

6.2.2 Message

In authentication phase, four types of messages are exchanged: REGISTER message, REGISTER response message, MESSAGE message and MESSAGE response message.

a) REGISTER message

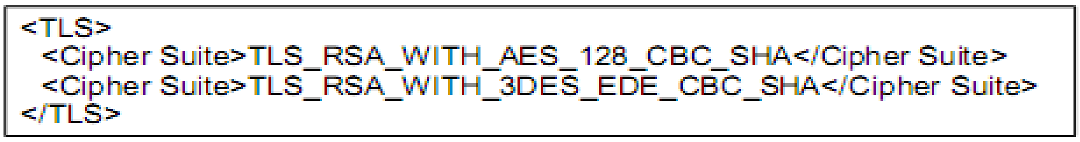
REGISTER message is sent from TLS client (TLS server) to SIP server.

b) REGISTER response message

Register response message is sent from SIP server to the sender of REGISTER. If SIP server accepts the REGISTER message, “200 OK” message is sent as REGISTER response message.

c) MESSAGE message

MESSAGE message is sent from TLS client (TLS server) to SIP server. The body of this message carries security policy:



```
<TLS>
  <Cipher Suite>TLS_RSA_WITH_AES_128_CBC_SHA</Cipher Suite>
  <Cipher Suite>TLS_RSA_WITH_3DES_EDE_CBC_SHA</Cipher Suite>
</TLS>
```

Pict3: Example of security policy in MESSAGE message during authentication phase

The security policy in this figure states that TLS client (or TLS server) has two available cipher suites for TLS session: one is the cipher suite “TLS_RSA_WITH_AES_128_CBC_SHA” and another is the cipher suite “TLS_RSA_WITH_3DES_EDE_CBC_SHA”

d) MESSAGE response message

MESSAGE response message is sent from SIP server to the sender of MESSAGE message. If SIP server accepts the MESSAGE message, “200 OK” message is sent as MESSAGE response message.

6.3 Key distribution phase

6.3.1 Message sequences and system behaviors

In key distribution phase, TLS client, TLS server and SIP server exchange SIP message via SIP sessions which are established during authentication phase.

Picture 4 shows message sequence during key distribution phase as follow:

(1) In picture 4, TLS client sends an INVITE message to SIP server to request TLS session establishment with TLS server at first.

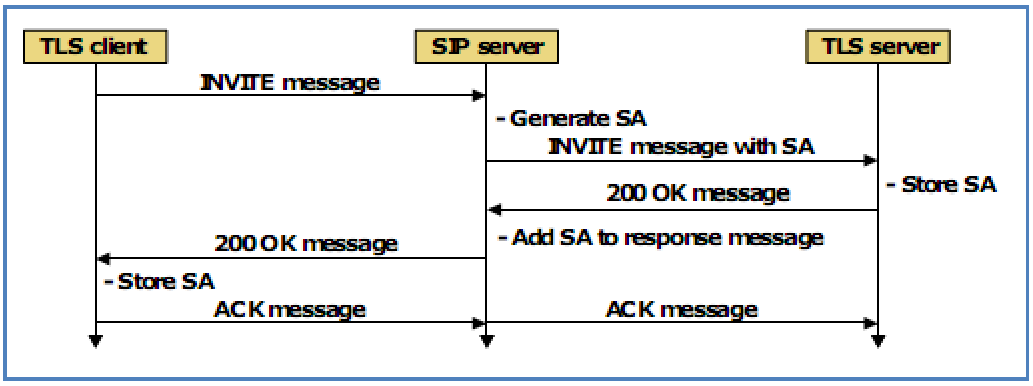
(2) SIP server, which receives INVITE message from TLS client, generates a security association from security policies of TLS client and TLS server. If no security association is generated (there is no available algorithms), SIP server returns 4xx response to TLS client. Otherwise, SIP server sends INVITE message, which contains the security association, to TLS server.

(3) TLS server that receives the INVITE message from SIP server stores the security association and returns a response message (200 OK).

(4) SIP server that receives the response message from TLS server adds the security association to the response message and sends the response message with the security association to TLS client.

(5) If TLS client that receives "200 OK" response message, TLS client stores the security association. Otherwise, TLS client disposes of the message.

(6) Finally, (if TLS client stores the security association successfully,) TLS client sends ACK message to TLS server via SIP server.



Pict4: Message sequence during key distribution phase

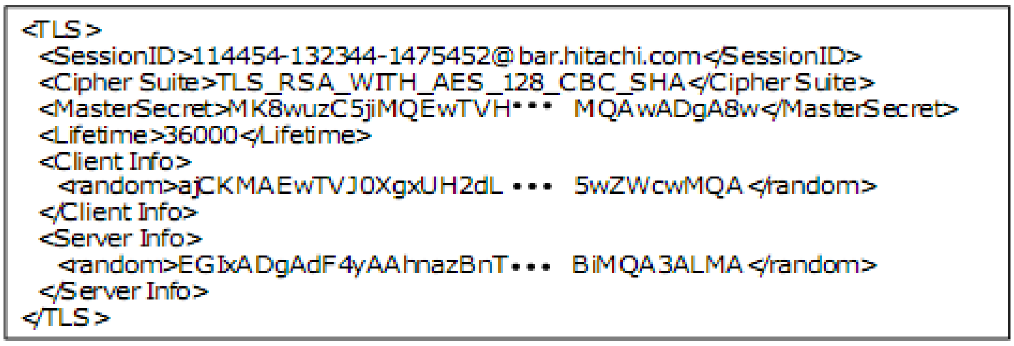
6.3.2 Messages

In key distribution phase, three types of messages are exchanged: INVITE message, INVITE response message and ACK message.

(a) INVITE message

INVITE message is sent from TLS client to TLS server via SIP server. This message is used to notify that TLS client ask TLS session establishment to TLS server via SIP server. The body of this message is used to carry security association from SIP server to TLS server like figure 5.

The security association in figure 5 is a candidate of security association. The security association contains a session ID that is wedged between "<sessionID>" and </sessionID>", a selected cipher suite that is wedged between "<cipherSuite>" and "</cipherSuite>", a master secret value that is wedged between "<masterSecret>" and "</masterSecret>", the life time of this security association that is wedged between "<lifetime>" and "</lifetime>", TLS client information for this session, that is wedged between "<clientInfo>" and "</clientInfo>" include a random value for this TLS session, and TLS server information for this session, that is wedged between "<serverInfo>" and "</serverInfo>" include a random value for this TLS session. TLS client information and TLS server information may contain IP address and port number used for this TLS session.



```
<TLS>
  <SessionID>114454-132344-1475452@bar.hitachi.com</SessionID>
  <Cipher Suite>TLS_RSA_WITH_AES_128_CBC_SHA</Cipher Suite>
  <MasterSecret>MK8wuzC5jiMQEwTVH•••  MQAwADgA8w</MasterSecret>
  <Lifetime>36000</Lifetime>
  <Client Info>
    <random>ajCKMAEwTVJ0XgxUH2dL •••  5wZWcwMQA</random>
  </Client Info>
  <Server Info>
    <random>EGIxADgAdF4yAAhnazBnT•••  BiMQA3ALMA</random>
  </Server Info>
</TLS>
```

Pict5: Example of security association in INVITE message during key distribution phase

(b) INVITE response message format

INVITE response message is sent from TLS server to TLS client via SIP server. If TLS server accepts the INVITE request, the status of this response message is "200 OK". The body of this message is also used to carry security association from SIP server to TLS client like picture 5.

(c) ACK message format

ACK message is sent from TLS client to TLS server via SIP server to notify that TLS client receive the INVITE response message to TLS server.

6.4 SA confirmation phase

6.4.1 Message sequences and system behaviors

In SA confirmation phase, TLS client and TLS server exchange TLS Handshake messages shown as picture 1. Namely, the following messages are exchanged between TLS client and TLS server.

(1) TLS client refers the security association shared at key distribution phase and sends Client Hello message with the session ID of security association to TLS server.

(2) TLS server, which receives Client Hello message, returns Server Hello message with same session ID to TLS client if TLS server can find the security association, which corresponds to the session ID.

(3) Then, TLS client sends Change Cipher Spec message and Finished message.

(4) TLS server sends Change Cipher Spec message and Finished message if TLS server can decrypt Finished message that received from TLS client.

(5) TLS client and TLS server transmit application data if TLS client can decrypt Finished message that received from TLS server.

6.4.2 Messages

In SA confirmation phase, four types of message are exchanged: Client Hello message, Server Hello message, Change Cipher Spec message and Finished message. These messages are compliant with RFC2246. And Client Hello message and Server Hello message contains the session ID of the security association shared at key distribution phase.

7 IPSEC IN VOIP NETWORKS

Psec is widely used to provide secure access to corporate private networks. IPsec is specified by TISPAN and 3GPP for both access, core and interconnect applications. However, IPsec and the current range of corporate network edge Network Address and Port Translators (NAPTs) are not compatible with the Next Gen services provided by VoIP architectures. This technical note describes the problems and the solution defined by TISPAN.

7.1 Background

IP does not have any in-built security capabilities, thus IPsec was introduced to provide the required security services. These include: encryption, authentication, integrity validation and anti-replay. IPsec operates at the network layer (layer 3) making it more flexible than TLS since it can encapsulate both UDP and TCP. However, IPsec assumes that the end-to-end connection does not have to traverse intermediate devices such as NATs which alter the authenticated packets.

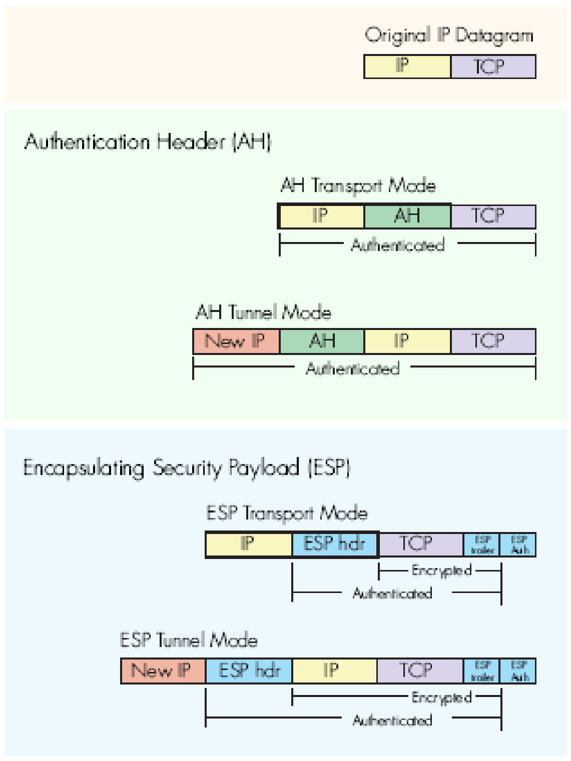
NATs are widely deployed throughout the business world (near 100% of all business networks) and also by many domestic IP users; since about 50% of domestic network edge devices have NAT functionality in them (e.g. ADSL modems). More correctly these devices should be call NAPTs – Network Address and Port Translators – unfortunately, as we shall see, IPsec obscures some of this information.

7.2 IPsec Details

IPsec has several different modes of operation: Authentication Header (AH) and Encapsulating Security Payload (ESP). These in turn have two connection modes, tunnel mode and transport mode. ETSI and 3GPP specify the use of IPsec ESP tunnel mode in TS 33.210 (interconnect and core) and ESP transport mode in TS 33.203 (access).

AH or ESP?

IPsec Authentication Header (AH) provides authentication of origin, message integrity checking and prevention of replay attacks. In AH mode there is no encryption of the payload, only the header is involved.



IPsec Encapsulating Security Payload (ESP] offers message integrity, data confidentiality, prevention of replay attacks and optionally authentication.

Transport or Tunnel?

In general transport mode is used to secure end-to-end communications between two devices, whilst tunnel mode is used to connect two networks. This is reflected in the 3GPP selection of ESP transport mode for access and tunnel for interconnect.

In transport mode IPsec AH protects both the payload and the IP header fields and inserts a new header between the original IP header and the payload. In tunnel mode the whole IP packet is encapsulated within an AH and new IP header.

The AH header allows the recipient to detect out of sequence packets, and authenticate the sender. It also protects the integrity of the payload and header, the recipient recalculates. The hash and a mismatch indicates data tampering or incorrect key, the packet it therefore discarded.

For this reason IPsec AH is incompatible with NATs of any type since the function of a NAT is to alter the packet header, specifically the IP address. The AH mechanism will then cause the packet to be discarded as the calculation of the data integrity will indicate that the packet has been altered.

ESP in both transport and tunnel modes encapsulates and encrypts the required data and appends it to the original IP header (only changing the protocol field), therefore ESP is more 'NAT friendly'. Transport mode ESP encapsulates just the payload, i.e. the UDP or TCP part, whilst tunnel mode encapsulates the whole IP packet. There is no check that the encrypted portion matches the non-encrypted portion, and so NATs can be readily traversed.

In order to provide a secure path for VoIP signalling applications it can be seen that ESP with authentication provides the required security and authentication combination. However, 3GPP have assumed this will operate in a NAT free network.

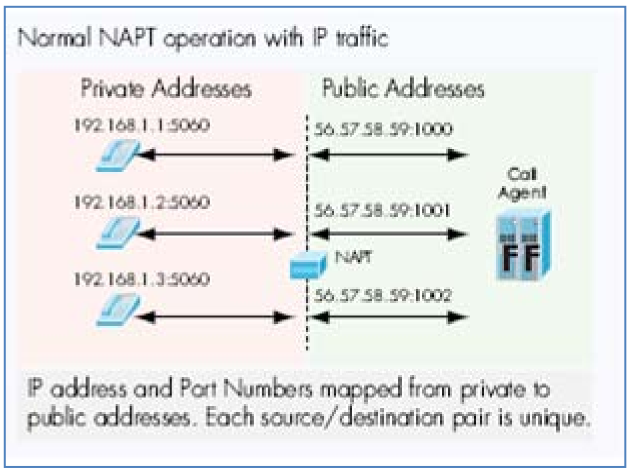
7.3 The Problem with IPsec and NATs

Most NAT devices are used to provide a mapping of many devices on a private network onto a single public IP address. They do this by mapping ports as well as IP addresses. This means they alter the contents of the transport layer protocol, i.e UDP or TCP. When a NAPT device encounters an IPsec ESP packet it no longer has access to the transport layer ports and will usually revert to NAT-only operation.

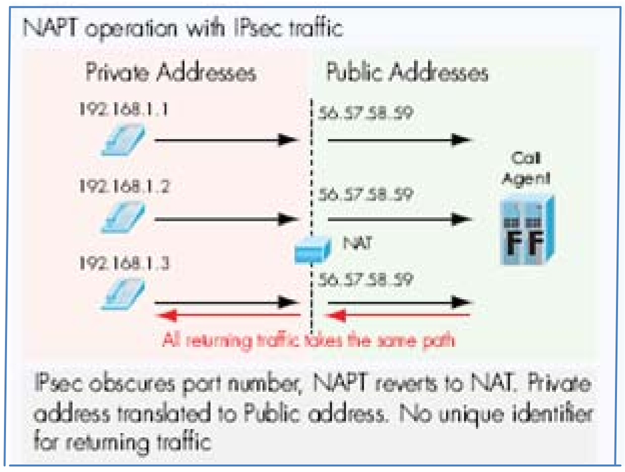
If there is only one IPsec device behind the NAT this is not a problem, since the NAT will simply translate the private IP address to the public IP address. This enables the single device to communicate through the NAT to the far end.

However, there is a problem where there are multiple IPsec sources behind a NAPT communicating with a single server. For example, several VoIP phones located on a business network talking to the same SIP server in the hosting Service Provider's network.

As we have seen, IPsec encapsulates and obscures the transport layer port information which the NAPT needs to create unique bindings between source IP address and port and destination IP address and port. This behaviour is more correctly termed symmetric Network Address and Port



Normal NAPT operation with IP traffic
Private Addresses
Public Addresses
192.168.1.1:5060
56.57.58.59:1000
192.168.1.2:5060
56.57.58.59:1001
192.168.1.3:5060
56.57.58.59:1002
NAPT
Call Agent
IP address and Port Numbers mapped from private to public addresses. Each source/destination pair is unique.



NAPT operation with IPsec traffic
Private Addresses
Public Addresses
192.168.1.1
56.57.58.59
192.168.1.2
56.57.58.59
192.168.1.3
56.57.58.59
NAT
Call Agent
All returning traffic takes the same path
IPsec obscures port number, NAPT reverts to NAT. Private address translated to Public address. No unique identifier for returning traffic

Translator, and is by far the most widely used NAPT mechanism in access networks.

Presented with a number of IPsec streams heading out to the same destination, “VPN compatible NAPTs” change to NAT mode, thus they create a many-to-one relationship in the bindings. So, if two IPsec phones are trying to access the same server from behind the same NAPT, there is no way for the returning traffic to be steered via the appropriate binding to the correct phone.

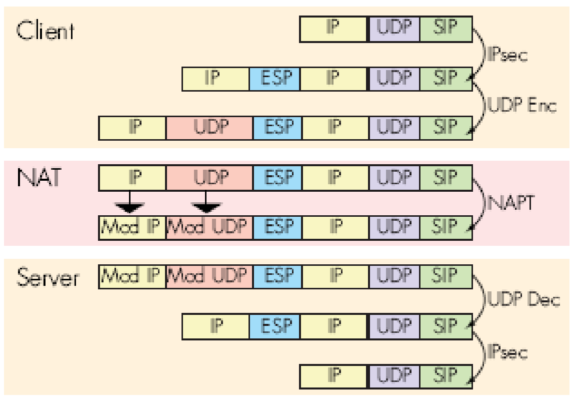
In practice most NAPT products with a "VPN compatibility" mode use the binding created by the last outbound packet as the destination for inbound packets. Thus, one user will receive all the signalling.

7.4 TISPAN Security

In a mobile 3GPP network IPsec presents no problem because there are no NAPT devices between the phone and the server. Similarly, using IPsec ESP in tunnel mode to interconnect networks is unlikely to have to traverse NAPTs. However, to

secure the signalling from phone to server in a broadband or Internet based network an alternative solution must be used.

TISPAN has recognised that NAT traversal is not possible with the IMS security solutions in TS 33.203 releases 5 and 6. It has agreed to use UDP encapsulated IPsec according to RFC 3948. TS 33.203 defines a security negotiation mechanism (RFC 3329) which includes a Mode parameter, TISPAN has extended the parameters to include "UDP-enc-tun". The encapsulation occurs in two stages: the original clear SIP/UDP message is encapsulated using IPsec ESP, then this encapsulated in another UDP header. The message is sent from port 4500 to port 4500. The intermediate NAPT device can change the source IP and port numbers without affecting the IPsec payload. De-encapsulation is the reverse process.



7.5 Conclusion

As NAPT devices are a fact of life for most broadband, WiFi and WiMax networks and IPsec is incompatible with NAPT it

has been necessary for TISPAN to extend the 3GPP security mechanisms to address this crucial need. The ability to negotiate a secure signalling connection with the IMS will reduce service theft (toll fraud), increase privacy and will build consumer confidence in the next generation of multimedia service. TISPAN's extension of the 3GPP security framework to include UDP encapsulated IPsec means that any number of intermediate NAT or NAPT devices can be traversed. The NAT devices have access to the encapsulating UDP port numbers and can perform address and port translation in the normal way. The use of a common standard for the negotiation and delivery of a secure signalling path will greatly improve device interoperability.

Since this is an extension of the existing IMS security framework it means that other mechanisms such as key exchange remain unchanged.

There are other advantages in using IPsec as opposed to the alternative which was under consideration – TLS. TLS sits above the transport layer so port numbers are still visible to intermediate NAPTs. However, typically only the server is authenticated, and the client remains unauthenticated. TLS demands the use of TCP, whereas IPsec can use either UDP or TCP. The use of TCP in large scale deployments raises concerns about the overhead of maintaining a large number of connections with numerous UAs. Thus, TCP is not well liked by service providers since the overheads associated with its mass use are significant compared to UDP.

TISPAN's extension to the IMS security framework delivers the capability of delivering secure signalling whilst maintaining the ability to traverse intermediate NAPT devices.

8 The Secure Real-Time Transport Protocol (SRTP)

Both RTP and RTCP packets can be cryptographically secured by the Secure Real-time Transport Protocol (SRTP) and the companion Secure Real-time Transport Control Protocol (SRTCP), respectively [Ba04]. This chapter gives the details on the message formats and sheds some light on session key generation and master key distribution.

8.1 The Secure RTP Packet Format

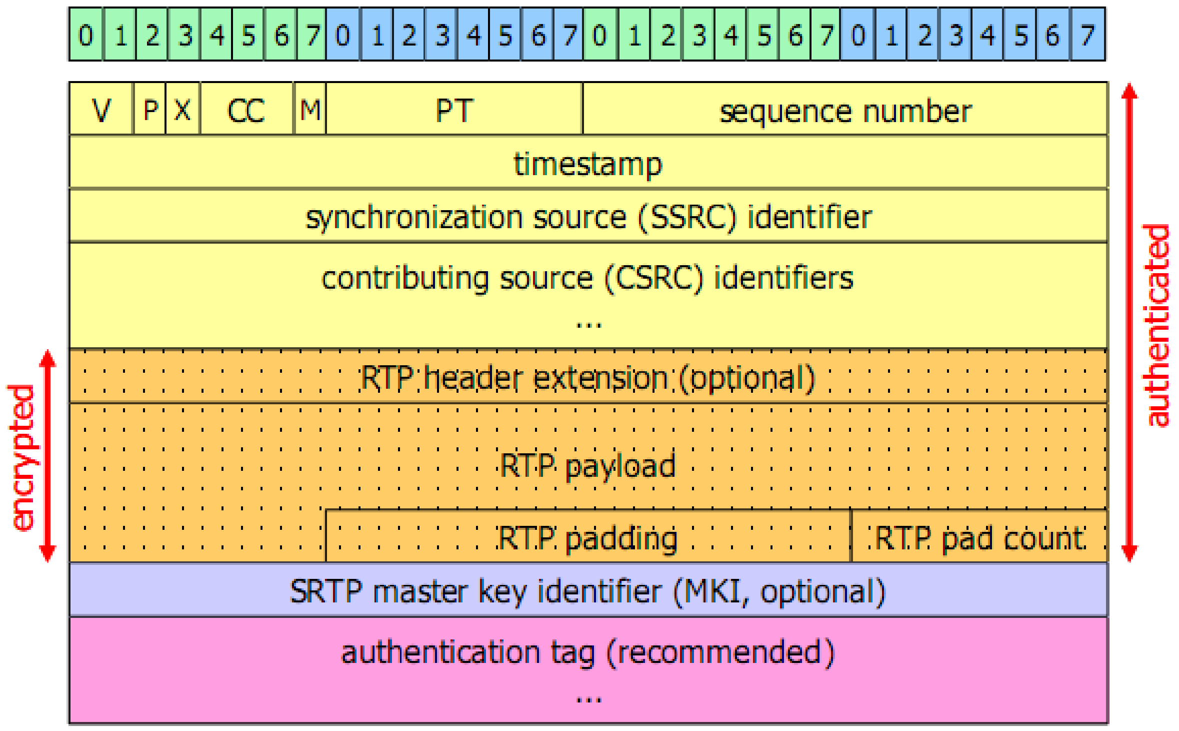

0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
V
P
X
CC
M
PT
sequence number
timestamp
synchronization source (SSRC) identifier
contributing source (CSRC) identifiers
...
RTP header extension (optional)
RTP payload
RTP padding
RTP pad count
SRTP master key identifier (MKI, optional)
authentication tag (recommended)
...
encrypted
authenticated

Figure 5: Secure RTP packet format

The SRTP message format is shown in Figure 5. As can be seen, only the RTP payload body (including any RTP padding if present) is encrypted. Since all currently defined encryption transforms do not add any padding, the size of the RTP payload is not increased by encryption.

The Master Key Identifier (MKI) field is optional and identifies the master key from which the session keys were derived. The MKI can be used by the receiver to retrieve the correct master key when the need for a re-keying event comes up. The 16 bit sequence number already present in the RTP packet is used together with a 32 bit rollover counter (ROC) which is part of the cryptographic context for the SRTP session to prevent replay attacks. The authentication tag is a cryptographic checksum computed over both the header and body of the RTP packet. Its use is strongly recommended since it protects the packets from unauthorized modification. The default tag length is 10 bytes but might be reduced if the transmission channel does not allow such a large increase of the RTP packet size.

8.2 The Secure RTCP Packet Format

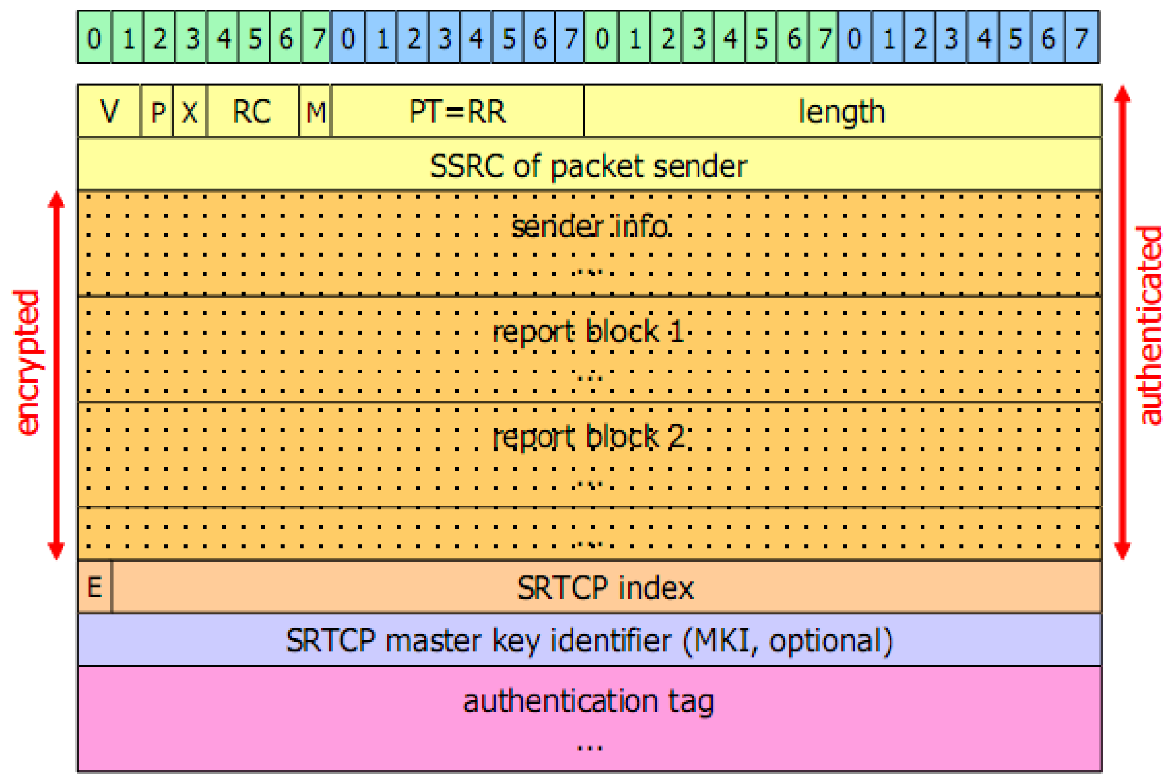


Figure 6: Secure RTCP packet format

Figure 6 shows that the RTP control packets are secured in a similar way as the RTP packets themselves, one difference being that the use of the authentication tag is mandatory. Otherwise it would be possible for a malevolent attacker e.g. to terminate an RTP media stream by sending a BYE packet. An additional field is the SRTCP index which used as a sequence counter preventing replay-attacks. The MSB of the index field is used as an Encryption flag (E) which is set if the RTCP body is encrypted.

8.3 Default Encryption Algorithms

In principle any encryption scheme can be used with SRTP. As default algorithms the NULL cipher (no confidentiality) and the Advanced Encryption Standard in Counter Mode (AES-CTR) are defined. The AES-CTR encryption setup is shown in Figure 7.

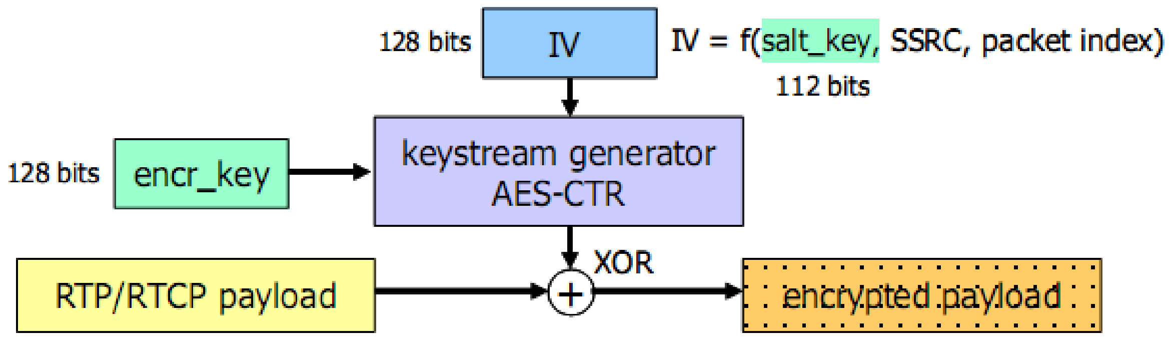


Figure 7: Encryption using AES in counter mode. 9

AES in counter mode acts as a keystream generator producing a pseudo-random key-stream of arbitrary length that is applied in a bit-wise fashion to the RTP/RTCP payload by means of a logical XOR function, thus working as a classical stream cipher. AES itself is a block cipher with a block size of 128 bits and a key size of 128, 192, or 256 bits. In order to work as a pseudo-random generator AES is loaded at the start of each RTP/RTCP packet with a distinct initialisation vector (IV) that is derived by hashing a 112 bit salt_key, the synchronisation source identifier (SSRC) of the media stream, and the packet index. Encrypting this IV results in an output of 128 pseudo-random bits. Next the IV is incremented by one and again encrypted, thus generating the next 128 bits of the keystream. By counting the IV up by increments of one as many keystream blocks can be generated as are required to encrypt the whole RTP/RTPC payload. Any remaining bits from the last keystream block are simply discarded. AES used in counter mode instead of the more common cipher block chaining mode (CBC) has the big advantage that the keystream can be precomputed before the payload becomes available thus minimizing the delay introduced by encryption. And of course by using a stream cipher instead of block cipher there is no need to pad the payload up to a multiple of the block size which would add 15 overhead bytes to the RTP/RTCP packet in the worst case.

Komunikácia dvoch klientov cez systém 3CX

```
3	192.168.0.4	192.168.0.2	SIP/SDP		Request: INVITE sip:100@192.168.0.2:5060,
with session description

Via: SIP/2.0/UDP 192.168.0.4:5060;branch=z9hG4bK-d8754z-5b7ebf5435766161-1---d8754z-;rport
Max-Forwards: 70
Contact: <sip:101@192.168.0.4:5060>
To: <sip:100@192.168.0.2:5060>
From: "3CXPhone"<sip:101@192.168.0.2:5060>;tag=9e1b2357
Call-ID: MDhkNTJjYWE0YzI0NzZkODQ0ODQ0NTJhYzY0ZDY3ZjA.
CSeq: 1 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REGISTER, SUBSCRIBE, NOTIFY, REFER, INFO
Content-Type: application/sdp
Supported: replaces
User-Agent: 3CX Phone 6.0.727.0
Content-Length: 266



4	192.168.0.2	192.168.0.4	SIP	Status: 407 Proxy Authentication Required

SIP/2.0 407 Proxy Authentication Required
Via: SIP/2.0/UDP 192.168.0.4:5060;branch=z9hG4bK-d8754z-5b7ebf5435766161-1---d8754z-
;rport=5060
Proxy-Authenticate: Digest
nonce="12873903071:fc286ea4a4c6d8bc4c77897392531df6",algorithm=MD5,realm="3CXPhoneSyst
em"
To: <sip:100@192.168.0.2:5060>;tag=3457a320
From: "3CXPhone"<sip:101@192.168.0.2:5060>;tag=9e1b2357
Call-ID: MDhkNTJjYWE0YzI0NzZkODQ0ODQ0NTJhYzY0ZDY3ZjA.
CSeq: 1 INVITE
User-Agent: 3CXPhoneSystem 7.0.4025.0
Content-Length: 0



5	192.168.0.4	192.168.0.2	SIP	Request: ACK sip:100@192.168.0.2:5060

Via: SIP/2.0/UDP 192.168.0.4:5060;branch=z9hG4bK-d8754z-5b7ebf5435766161-1---d8754z-;rport
Max-Forwards: 70
To: <sip:100@192.168.0.2:5060>;tag=3457a320
From: "3CXPhone"<sip:101@192.168.0.2:5060>;tag=9e1b2357
Call-ID: MDhkNTJjYWE0YzI0NzZkODQ0ODQ0NTJhYzY0ZDY3ZjA.
CSeq: 1 ACK
Content-Length: 0



6	192.168.0.4	192.168.0.2	SIP/SDP		Request: INVITE sip:100@192.168.0.2:5060,
with session description

Via: SIP/2.0/UDP 192.168.0.4:5060;branch=z9hG4bK-d8754z-2f6fac45d378d23b-1---d8754z-;rport
Max-Forwards: 70
Contact: <sip:101@192.168.0.4:5060>
To: <sip:100@192.168.0.2:5060>
```

```
From: "3CXPhone"<sip:101@192.168.0.2:5060>;tag=9e1b2357
Call-ID: MDhkNTJjYWE0YzI0NzZkODQ0ODQ0NTJhYzY0ZDY3ZjA.
CSeq: 2 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REGISTER, SUBSCRIBE, NOTIFY, REFER, INFO
Content-Type: application/sdp
Proxy-Authorization: Digest

username="101",realm="3CXPhoneSystem",nonce="12873903071:fc286ea4a4c6d8bc4c7789739253
1df6",uri="sip:100@192.168.0.2:5060",res

ponse="7c4644322e1063ce37f3e35381a3844d",algorithm=MD5
Supported: replaces
User-Agent: 3CX Phone 6.0.727.0
Content-Length: 266



7 192.168.0.2    192.168.0.3       SIP/SDP           Request: INVITE
sip:100@192.168.0.3:5060;rinstance=ef24715eed5fb03d, with session description

Via: SIP/2.0/UDP 192.168.0.2:5060;branch=z9hG4bK-d8754z-d62ee84d2a103c4c-1---d8754z-;rport
Max-Forwards: 70
Contact: <sip:101@192.168.0.2:5060>
To: "3CXPhone"<sip:100@192.168.0.2:5060>
From: "3CXPhone"<sip:101@192.168.0.2:5060>;tag=8a3c625f
Call-ID: ZWMxYzY4MWUyZDZlZjU1MzQwZTQxNWFjODU1NzQ2OTg.
CSeq: 1 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REGISTER, SUBSCRIBE, NOTIFY, REFER, INFO
Content-Type: application/sdp
User-Agent: 3CXPhoneSystem 7.0.4025.0
Content-Length: 270



8          192.168.0.2       192.168.0.4       SIP       Status: 100 Trying

Via: SIP/2.0/UDP 192.168.0.4:5060;branch=z9hG4bK-d8754z-2f6fac45d378d23b-1---d8754z-
;rport=5060
To: <sip:100@192.168.0.2:5060>
From: "3CXPhone"<sip:101@192.168.0.2:5060>;tag=9e1b2357
Call-ID: MDhkNTJjYWE0YzI0NzZkODQ0ODQ0NTJhYzY0ZDY3ZjA.
CSeq: 2 INVITE
Content-Length: 0



9          192.168.0.2       192.168.0.4       SIP/XML           Request: NOTIFY sip:101@192.168.0.4:5060

Via: SIP/2.0/UDP 192.168.0.2:5060;branch=z9hG4bK-d8754z-c91ce80738090d0e-1---d8754z-;rport
Max-Forwards: 70
Contact: <sip:%23any-dn%23@192.168.0.2:5060>
To: "3CXPhone"<sip:101@192.168.0.2:5060>;tag=56224c4c
From: "3CXPhone"<sip:%23any-dn%23@192.168.0.2:5060>;tag=57584659
Call-ID: ZDY2NWFmMzc4NDA2NTA5NTU0MGY3NjZlMjQzMzI1Zjk.
CSeq: 36 NOTIFY
Content-Type: application/pidf+xml
User-Agent: 3CXPhoneSystem 7.0.4025.0
Subscription-State: active;expires=46
Event: presence
```

```
Content-Length: 283



10      192.168.0.2     192.168.0.3     SIP/XML         Request: NOTIFY sip:100@192.168.0.3:5060

Via: SIP/2.0/UDP 192.168.0.2:5060;branch=z9hG4bK-d8754z-7d0ad625103ff056-1---d8754z-;rport
Max-Forwards: 70
Contact: <sip:%23any-dn%23@192.168.0.2:5060>
To: "3CXPhone"<sip:100@192.168.0.2:5060>;tag=c85ce30f
From: "3CXPhone"<sip:%23any-dn%23@192.168.0.2:5060>;tag=32391b41
Call-ID: YjE4ZWJhZGMxYjU5OWFlNzgzYjY3ZjY5NzI0ZDQzMTU.
CSeq: 25 NOTIFY
Content-Type: application/pidf+xml
User-Agent: 3CXPhoneSystem 7.0.4025.0
Subscription-State: active;expires=78
Event: presence
Content-Length: 283



11      192.168.0.4     192.168.0.2     SIP     Status: 200 OK

Via: SIP/2.0/UDP 192.168.0.2:5060;branch=z9hG4bK-d8754z-c91ce80738090d0e-1---d8754z-
;rport=5060
Contact: <sip:101@192.168.0.4:5060>
To: "3CXPhone"<sip:101@192.168.0.2:5060>;tag=56224c4c
From: "3CXPhone"<sip:%23any-dn%23@192.168.0.2:5060>;tag=57584659
Call-ID: ZDY2NWFmMzc4NDA2NTA5NTU0MGY3NjZlMjQzMzI1Zjk.
CSeq: 36 NOTIFY
User-Agent: 3CX Phone 6.0.727.0
Content-Length: 0



12      192.168.0.3     192.168.0.2     SIP     Status: 180 Ringing

Via: SIP/2.0/UDP 192.168.0.2:5060;branch=z9hG4bK-d8754z-d62ee84d2a103c4c-1---d8754z-
;rport=5060
Contact: <sip:100@192.168.0.3:5060;rinstance=ef24715eed5fb03d>
To: "3CXPhone"<sip:100@192.168.0.2:5060>;tag=bb64467f
From: "3CXPhone"<sip:101@192.168.0.2:5060>;tag=8a3c625f
Call-ID: ZWMxYzY4MWUyZDZlZjU1MzQwZTQxNWFjODU1NzQ2OTg.
CSeq: 1 INVITE
User-Agent: 3CX Phone 6.0.727.0
Content-Length: 0



13      192.168.0.3     192.168.0.2     SIP     Status: 200 OK

Via: SIP/2.0/UDP 192.168.0.2:5060;branch=z9hG4bK-d8754z-7d0ad625103ff056-1---d8754z-
;rport=5060
Contact: <sip:100@192.168.0.3:5060>
To: "3CXPhone"<sip:100@192.168.0.2:5060>;tag=c85ce30f
From: "3CXPhone"<sip:%23any-dn%23@192.168.0.2:5060>;tag=32391b41
Call-ID: YjE4ZWJhZGMxYjU5OWFlNzgzYjY3ZjY5NzI0ZDQzMTU.
CSeq: 25 NOTIFY
User-Agent: 3CX Phone 6.0.727.0
```

```
Content-Length: 0



14        192.168.0.2       192.168.0.4       SIP       Status: 180 Ringing

Via: SIP/2.0/UDP 192.168.0.4:5060;branch=z9hG4bK-d8754z-2f6fac45d378d23b-1---d8754z-
;rport=5060
Contact: <sip:101@192.168.0.2:5060>
To: <sip:100@192.168.0.2:5060>;tag=9d21b244
From: "3CXPhone"<sip:101@192.168.0.2:5060>;tag=9e1b2357
Call-ID: MDhkNTJjYWE0YzI0NzZkODQ0ODQ0NTJhYzY0ZDY3ZjA.
CSeq: 2 INVITE
User-Agent: 3CXPhoneSystem 7.0.4025.0
Content-Length: 0



15        192.168.0.2       192.168.0.4       SIP/XML          Request: NOTIFY sip:101@192.168.0.4:5060

Via: SIP/2.0/UDP 192.168.0.2:5060;branch=z9hG4bK-d8754z-8c652a5a852a6600-1---d8754z-;rport
Max-Forwards: 70
Contact: <sip:%23any-dn%23@192.168.0.2:5060>
To: "3CXPhone"<sip:101@192.168.0.2:5060>;tag=56224c4c
From: "3CXPhone"<sip:%23any-dn%23@192.168.0.2:5060>;tag=57584659
Call-ID: ZDY2NWFmMzc4NDA2NTA5NTU0MGY3NjZlMjQzMzI1Zjk.
CSeq: 37 NOTIFY
Content-Type: application/pidf+xml
User-Agent: 3CXPhoneSystem 7.0.4025.0
Subscription-State: active;expires=46
Event: presence
Content-Length: 283



16        192.168.0.2       192.168.0.3       SIP/XML          Request: NOTIFY sip:100@192.168.0.3:5060

Via: SIP/2.0/UDP 192.168.0.2:5060;branch=z9hG4bK-d8754z-f04c59699011e43b-1---d8754z-;rport
Max-Forwards: 70
Contact: <sip:%23any-dn%23@192.168.0.2:5060>
To: "3CXPhone"<sip:100@192.168.0.2:5060>;tag=c85ce30f
From: "3CXPhone"<sip:%23any-dn%23@192.168.0.2:5060>;tag=32391b41
Call-ID: YjE4ZWJhZGMxYjU5OWFlNzgzYjY3ZjY5NzI0ZDQzMTU.
CSeq: 26 NOTIFY
Content-Type: application/pidf+xml
User-Agent: 3CXPhoneSystem 7.0.4025.0
Subscription-State: active;expires=78
Event: presence
Content-Length: 283



17        192.168.0.3       192.168.0.2       SIP       Status: 200 OK

Via: SIP/2.0/UDP 192.168.0.2:5060;branch=z9hG4bK-d8754z-f04c59699011e43b-1---d8754z-
;rport=5060
Contact: <sip:100@192.168.0.3:5060>
To: "3CXPhone"<sip:100@192.168.0.2:5060>;tag=c85ce30f
From: "3CXPhone"<sip:%23any-dn%23@192.168.0.2:5060>;tag=32391b41
```

```
Call-ID: YjE4ZWJhZGMxYjU5OWFlNzgzYjY3ZjY5NzI0ZDQzMTU.
CSeq: 26 NOTIFY
User-Agent: 3CX Phone 6.0.727.0
Content-Length: 0

.
.
.
.
.
.
.
.
.



78      192.168.0.4      192.168.0.2      SIP      Request: BYE sip:101@192.168.0.2:5060

Via: SIP/2.0/UDP 192.168.0.4:5060;branch=z9hG4bK-d8754z-b21c800f7f4f1205-1---d8754z-;rport
Max-Forwards: 70
Contact: <sip:101@192.168.0.4:5060>
To: <sip:100@192.168.0.2:5060>;tag=9d21b244
From: "3CXPhone"<sip:101@192.168.0.2:5060>;tag=9e1b2357
Call-ID: MDhkNTJjYWE0YzI0NzZkODQ0ODQ0NTJhYzY0ZDY3ZjA.
CSeq: 3 BYE
Proxy-Authorization: Digest
username="101",realm="3CXPhoneSystem",nonce="12873903071:fc286ea4a4c6d8bc4c7789739253
1df6",uri="sip:101@192.168.0.2:5060",response="94f66473b4cae45eda0b90c4dfa4b825",algorithm=
MD5
User-Agent: 3CX Phone 6.0.727.0
Content-Length: 0



79      192.168.0.2      192.168.0.4      SIP      Status: 200 OK

Via: SIP/2.0/UDP 192.168.0.4:5060;branch=z9hG4bK-d8754z-b21c800f7f4f1205-1---d8754z-
;rport=5060
Contact: <sip:101@192.168.0.2:5060>
To: <sip:100@192.168.0.2:5060>;tag=9d21b244
From: "3CXPhone"<sip:101@192.168.0.2:5060>;tag=9e1b2357
Call-ID: MDhkNTJjYWE0YzI0NzZkODQ0ODQ0NTJhYzY0ZDY3ZjA.
CSeq: 3 BYE
User-Agent: 3CXPhoneSystem 7.0.4025.0
Content-Length: 0



80      192.168.0.2      192.168.0.3      SIP      Request: BYE

sip:100@192.168.0.3:5060;rinstance=ef24715eed5fb03d

Via: SIP/2.0/UDP 192.168.0.2:5060;branch=z9hG4bK-d8754z-087efd0f2d5cf60f-1---d8754z-;rport
Max-Forwards: 70
Contact: <sip:101@192.168.0.2:5060>
To: "3CXPhone"<sip:100@192.168.0.2:5060>;tag=bb64467f
From: "3CXPhone"<sip:101@192.168.0.2:5060>;tag=8a3c625f
Call-ID: ZWMxYzY4MWUyZDZlZjU1MzQwZTQxNWFjODU1NzQ2OTg.
```

```
CSeq: 2 BYE
User-Agent: 3CXPhoneSystem 7.0.4025.0
Content-Length: 0



83        192.168.0.2      SIP       Status: 200 OK

Via: SIP/2.0/UDP 192.168.0.2:5060;branch=z9hG4bK-d8754z-087efd0f2d5cf60f-1---d8754z-
;rport=5060
Contact: <sip:100@192.168.0.3:5060;rinstance=ef24715eed5fb03d>
To: "3CXPhone"<sip:100@192.168.0.2:5060>;tag=bb64467f
From: "3CXPhone"<sip:101@192.168.0.2:5060>;tag=8a3c625f
Call-ID: ZWMxYzY4MWUyZDZlZjU1MzQwZTQxNWFjODU1NzQ2OTg.
CSeq: 2 BYE
User-Agent: 3CX Phone 6.0.727.0
Content-Length: 0
```

10 Použitá literatúra

http://www.ietf.org/rfc/rfc3261.txt
http://en.wikipedia.org/wiki/Session_Initiation_Protocol
http://www.ietf.org/rfc/rfc3853.txt
www.site.uottawa.ca/~bob/gradstudents/DigestAuthenticationReport.pdf
security.hsr.ch/docs/DFN_SIP.pdf
www.voip-info.org
www.ietf.o
download.securelogix.com/librarydownload.htm?
downloadfilename=SIP_Security030105.pdfg/proceedings/01dec/slides/sip-7/index.html
phoenix.labri.fr/documentation/sip/Documentation/Papers/Security/Presentations/pres-
schulzrinne-SIP-Security-ietf52.ppt