Secure SIP between IPv4 Endpoints and IPv6 Endpoints

XING JIANG

A THESIS

IN

THE DEPARTMENT

OF

COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE AT
CONCORDIA UNIVERSITY
MONTREAL, QUEBEC, CANADA

DECEMBER. 2003

© XING JIANG, 2003

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisisitons et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-91052-0
Our file Notre référence
ISBN: 0-612-91052-0

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this dissertation.

While these forms may be included in the document page count, their removal does not represent any loss of content from the dissertation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de ce manuscrit.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Secure SIP between IPv4 Endpoints and IPv6 Endpoints

Xing Jiang

The Internet Protocol version 6 (IPv6) is designed to replace the current version IPv4. However, there will continue to be a demand for IPv4-based end users to access IPv6-based services, and vice versa. Some transition mechanisms are necessary to make IPv6 and IPv4 compatible. Network Address Translation - Protocol Translation (NAT-PT) can provide protocol translation at the network layer.

The Session Initiation Protocol (SIP) is an application layer control protocol that can initiate, modify and terminate interactive communication sessions between end users. When SIP is used with NAT-PT, a special Application Level Gateway (ALG) is required to handle the translation of the addresses inside the SIP messages.

This thesis introduces an implementation of a SIP-ALG. The SIP-ALG is responsible for translating IPv6 addresses in a SIP packet into the corresponding IPv4 addresses, and vice versa, relying on the functionalities of NAT-PT as the packet traverses across the boundary between IPv6 and IPv4.

In addition, this thesis describes and models a SIP end-to-end security solution between IPv4 end points and IPv6 end points, given that involvement of the SIP-ALG seems to be in conflict with the primary requirements of the end-to-end security. The proposed mechanism lets a SIP endpoint authorize a security proxy server to encrypt the SIP bodies on behalf of the end point. The security proxy will discover the capabilities of the receiving party and encrypt the SIP bodies for the other SIP security proxy server in the receiving domain. IP address translation must be done before the encryption at the sending security proxy or after the decryption at the receiving security proxy.

Acknowledgements

Great thanks must go to my supervisor, Dr. John William Atwood, for his efficient guidance throughout the duration of my research. The Chinese have a proverb, "teaching fishing is better than giving fish". What he has been teaching me is not only the knowledge itself, but also, more importantly, the skills for doing research.

I also want to thank the instructors who taught me at Concordia University. The knowledge I obtained from their courses has proved to be very valuable as I continue my studies.

I would also like to acknowledge my parents and my sister, who have encouraged and supported me during my graduate studies. Many thanks should be given to my friend, Yinghong Chen, who provided me with a great deal of equipment for my research.

I wish to thank Ericsson Research Canada for their technical support and for the offer of using their laboratory.

Table of Contents

List of Figures

List of Tables

List of Acronyms

3GPP	Third Generation Partnership Project
AH	Authentication Header
ALG	Application Level Gateway
BNF	Backus-Naur Form
CMS	Cryptographic Message Syntax
CN	Common Name
DNS	Domain Name System
DNS-ALG	Domain Name System - Application Level Gateway
DoS	Denial-of-Service
ESP	Encapsulation Security Payload
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IPSec	Internet Protocol Security
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
MEGACO	Media Gateway Control Protocol
MIDCOM	Middlebox Communications protocol
MIME	Multipurpose Internet Mail Extensions
MitM	Man-in-the-Middle
MSC	Message Sequence Chart
NAT	Network Address Translation
NAPT-PT	Network Address Port Translation - Protocol Translation
NAT-PT	Network Address Translation - Protocol Translation
Promela	Process Meta Language

PSTN	Public Switched Telephone Network
RSTP	Real-Time Streaming Protocol
RTCP	RTP control protocol
RTP	Realtime Transport Protocol
SA	Security Association
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SIP-ALG	Session Initiation Protocol - Application Level Gateway
S/MIME	Secure Multipurpose Internet Mail Extension
SMTP	Simple Mail Transfer Protocol
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
VoIP	Voice over IP

1 Introduction

The Internet Protocol Version 4 (IPv4) is one of the most popular protocols in use today. However, IPv4 is more than 30 years old and has started to show its age. There are some questions about its capability to serve the Internet community any longer. The main issue is the lack of addressing. The Internet Protocol Version 6 (IPv6) is the next generation protocol designed by the Internet Engineering Task Force (IETF) to replace the current IPv4. For example, the Third Generation Partnership Project (3GPP), which is responsible for the standardization of the Third Generation (3G) mobile networks, has employed the Session Initiation Protocol (SIP) as the call control protocol and IPv6 as the only network protocol for 3G IP-based wireless networks.

"SIP is an application-layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls" [RFC3261]. SIP establishes sessions by its invitations in which session descriptions are used to negotiate a set of compatible media types to be shared among participants.

However, an obvious problem is that IPv6 is not compatible with IPv4. In addition, the schedule for completely upgrading the current IPv4 protocol to IPv6 is hard to predict; therefore, communication issues between the legacy IPv4 and the new IPv6 equipment must be solved. It is assumed that sessions between IPv4-only SIP clients and IPv6-only 3G mobile terminals will be needed; thus, some transition mechanisms are necessary to achieve the inter-working.

There was no document describing detailed implementation of the IPv4 and IPv6 inter-working with SIP protocol when I began working in this field, though, some experts have offered some good indications of how such an implementation might work. For example, the Network Address Translation – Protocol Translation (NAT-PT) [RFC2766] has been standardized as a way of connecting hosts in the IPv4 address space and hosts in the IPv6 address space. Together with NAT-PT, an Application Level Gateway (ALG) is able to alter both IP headers and IP payloads in order to translate SIP packets from the IPv4 Internet to IPv6 stub domain, and vice versa.

Even though the inter-working between the IPv4 and IPv6 application could work well by the proposal about coupling NAT-PT with a special ALG, more complicated issues arise. Network security, especially end-to-end security mechanism provides integrity, privacy and confidentiality of packets, that is, the packets must not be altered or viewed when they are transmitted between two end points. It is clear that the modification of SIP packets as they pass across the boundary between IPv4 domain and IPv6 network violates the requirements of the end-to-end network security mechanism. An Internet draft written by K. Umschaden [Umschaden03] introduces a solution of SIP end-to-end security in the NAT (IPv4-IPv4) environment in order to help solve this conflict. As of yet, however, there is no document that provides a complete solution to the SIP end-to-end security when the SIP sessions have to be established between an IPv6 stub domain and the IPv4 Internet.

Given the above, the goals of this thesis are:

- To understand the requirements on SIP needed for an IPv6 and address conversion; propose an architecture to meet those requirements; and to use modeling/implementation to verify that the architecture is viable.
- To propose an architecture for end-to-end secure SIP between IPv6 and IPv4 endpoints, and model/implement it for verification.

By the end of doing the research for this thesis, I already implemented a working SIP-ALG based on the implementations of the SIP user agents [Vocal03] and the NAT-PT modules [Natpt01]. Also, this implementation is part of a co-research project with Kedar Das at Ericsson Research Canada where Kedar installs and configures the NAT-PT modules. The SIP-ALG was written in C++ language on RedHat Linux version 7.3. Moreover, this work has been discussed in two papers published in 2003: "Establishing Multimedia Sessions between IP version 6 Hosts and IP version 4 Hosts." [Atwood03-1] and "IPv4/IPv6 Translation." [Atwood03-2]. Furthermore, I have designed an end-to-end SIP security architecture that extends the SIP protocol to secure SIP messages end-to-end by a proposed SIP security proxy server. Lastly, I have used the modeling tools, Promela/Spin, to simulate and verify the architecture.

In this thesis, I will begin by introducing three important protocols: Session Initiation Protocol (SIP), Session Description Protocol (SDP) and Realtime Transport Protocol (RTP), which are used together to set up IP telephony sessions. Chapter 3 describes Network Address Translation – Protocol Translation (NAT-PT) and Application Level

Gateway (ALG) with regards to IP addressing issues. In Chapter 4, my implementation, a SIP-ALG, is explained in detail. Chapter 5 provides an overview of potential threats to SIP and relevant SIP security mechanisms recommended by [RFC3261]. Chapter 6 is a detailed explanation about the proposed SIP end-to-end security mechanism. Chapter 7 introduces the simulation and verification of the security mechanisms proposed by both Umschaden and this thesis paper. The thesis concludes by introducing some further applications based on this paper.

2 SIP, SDP and RTP

This chapter will introduce the required protocols to make IP telephony work. To build a SIP communication some protocols must be recruited, typically the Session Description Protocol (SDP) for describing multimedia sessions and the Real-time Transport Protocol (RTP) for transporting real-time data and providing QoS feedback.

The introduction will start from low-level protocol towards high-level protocol. Section 2.1 looks at the Real-Time Protocol (RTP), which enables real-time audio transfer on the Internet. In Section 2.2, I will explain the Session Description Protocol (SDP), which explains to senders of real-time data how to encode audio in such a way that the recipient can decode and listen to the audio. Finally, in Section 2.3, I will discuss the Session Initiation Protocol (SIP), which enables users to find and contact each other in an organized, efficient manner.

2.1 RTP

Multimedia content has opened up a new world to the computer industry. This kind of data has expanded the applications of computer technology in fields such as education, entertainment and telecommunications.

The computer network industry continues to face the challenge of transporting this multimedia information over existing networks. Meeting such a challenge requires finding a solution for the following issues:

- High bandwidth needs (generally) for multimedia information
- Time-sensitivity of multimedia information
- Assurance of multicast delivery of information
- Carrying of as many types of multimedia content as possible

The efforts of the Internet Engineering Task Force (IETF) in this field led to the publication of the Real-time Transport Protocol (RTP) standard in 1996 [RFC1889]. Nowadays, this protocol is among the most common for multimedia information transport over the Internet. More than 20 payload types (the type of information that is being carried inside the packet) have been defined, and networking hardware manufacturers (such as Cisco) have included special handling features for RTP packet handing in the router software [Madrid00].

RTP provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video. [RFC1889] consists of two closely-linked protocols:

- The real-time transport protocol (RTP), to carry data that has real-time properties.
- The RTP control protocol (RTCP), to monitor the quality of service and to convey information about the participants in an on-going session.

2.1.1 RTP

The services provided by RTP are the following [Madrid00]:

- Payload type identification: Let applications using the protocol know which kind of information is being transmitted (audio, video, etc).

- Sequencing: When RTP is used over a datagram-based service, packet sequence numbers are required to reassemble the information stream in the destination.
- Timestamping: In applications such as teleconference, timestamps must be provided in the packets to assure synchronization of the devices participating in the conference; otherwise, information might be lost. This timestamping is also important for the controlling and monitoring of applications, and for having an accurate recording of the monitored events.
- Delivery monitoring: Allows the participating applications to collect and share statistics about the performance of the data transport service.

Furthermore, RTP is capable for using the multicast mechanism in the networks in order to establish one-to-many or many-to-many connections. In the Internet, RTP normally runs on top of the UDP protocol and makes use of its multiplexing and checksum services. However, any lower level protocol adequate for the application may be used.

2.1.1.1 RTP Fixed Header Fields

The header of the RTP packet has the following format [RFC1889, Madrid00]:

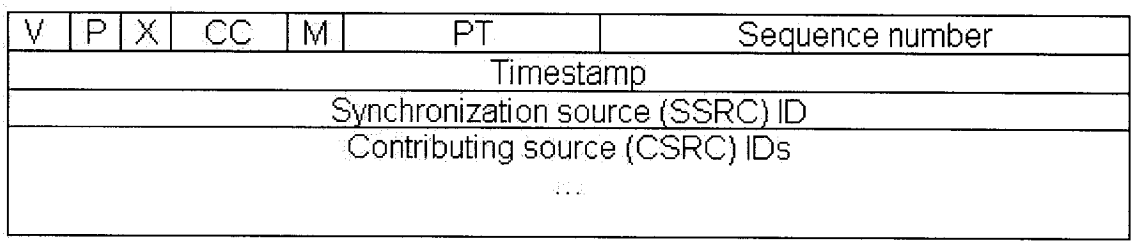

Figure 2-1 RTP Header Structure

The header fields have the following meaning:

- **V**: (2 bits) Version of protocol. Currently, version 2 of the protocol is used.
- **P**: (1 bit) Padding bit. If this bit is set, the packet contains padding bytes at the end. Padding is optional, and may be used for adjusting the RTP packet size to the size required by a lower layer protocol. The last byte of the packet tells how many padding bytes are in the packet.
- **X**: (1 bit) Extension bit. If this bit is set, the basic header is followed by a 32-bit extension, which can be used to extend the functionality of the applications.
- **CC**: (4 bits) CSRC count. CSRC stands for Contributing Source. When a set of RTP streams is processed by a mixer, the mixer inserts information about the original sources of the information in the headers of the RTP output stream. If the RTP stream has not been processed by a mixer, the CSRC count field shows zero; otherwise, it contains the number of mixed sources.
- **M**: (1 bit) Marker bit. The application using RTP defines the meaning of this marker bit, and when a packet is marked. This bit could be used, for example, to mark the start packet of a video frame.
- **PT**: (7 bits) Payload type. Specifies the kind of information the RTP packet is carrying.
- **Sequence number**: (16 bits) This number increases by one for each RTP packet sent. The receiver uses it to recover the original sequence of the packets and/or to determine if there are lost packets. The initial sequence number is random in

order to make decryption of the packets more difficult, in the case that the data stream is encrypted somewhere in the communication process.

- **Timestamp**: (32 bits) This field stores the sampling instant of the first byte in the packet payload. This field is used to synchronize different data streams and to extract quality of service statistics.
- **SSRC (Synchronization source)**: (32 bits) This randomly chosen number identifies the source of the RTP packets. It must be unique so all the data streams of a certain source may be clearly identified for decoding and playback on the receiver (each SSRC maintains its own counters for sequence numbers and timing, so it is very important that this number is unique). Once a node chooses a SSRC, it verifies via RTCP to see if the chosen number is unique.
- **CSRC list:** (0 to 15 items, 32 bits each) If the CSRC count is greater than zero, the identifiers for the contributing sources of this packet are listed. There is room for up to 15 identifiers; if there are more than 15 contributors only the first 15 are listed.

Note that the RTP header and its payload do not contain any information about IP addresses and TCP/UDP port numbers; therefore, RTP packets do not need to be modified even though their IP headers would be altered properly when they traverse a boundary between IPv4 domain and IPv6 domain.

2.1.2 RTCP

The RTP Control Protocol (RTCP) is used by nodes participating in a RTP communication to exchange statistical, user, and control information about the connection. It is based on the periodic transmission of control packets to all participants in the session, using the same distribution mechanism as the RTP data packets.

This control connection is held over a separate channel (uses another port number than the RTP one). Any given RTP media stream is always assigned to an even port number while the associated RTCP reports are sent on the next higher, and thus odd, port number.

2.1.3 RTP Use Scenario

Below are the typical processes whereby an RTP packet is sent to the Data Link Layer [Thernelius00].

1. The voice or video signals will be sampled at certain periods of time. A number of samples are bundled together by the application to be the payload encapsulated in a RTP packet. Typically 20 ms of sound is encapsulated into one RTP packet.
2. On the transportation level, the RTP packet is encapsulated into a UDP datagram.
3. On the network layer, the UDP datagram is encapsulated into an IP packet.
4. The IP packet is encapsulated into an Ethernet (or any other link layer protocol) frame and then the frame is sent off.

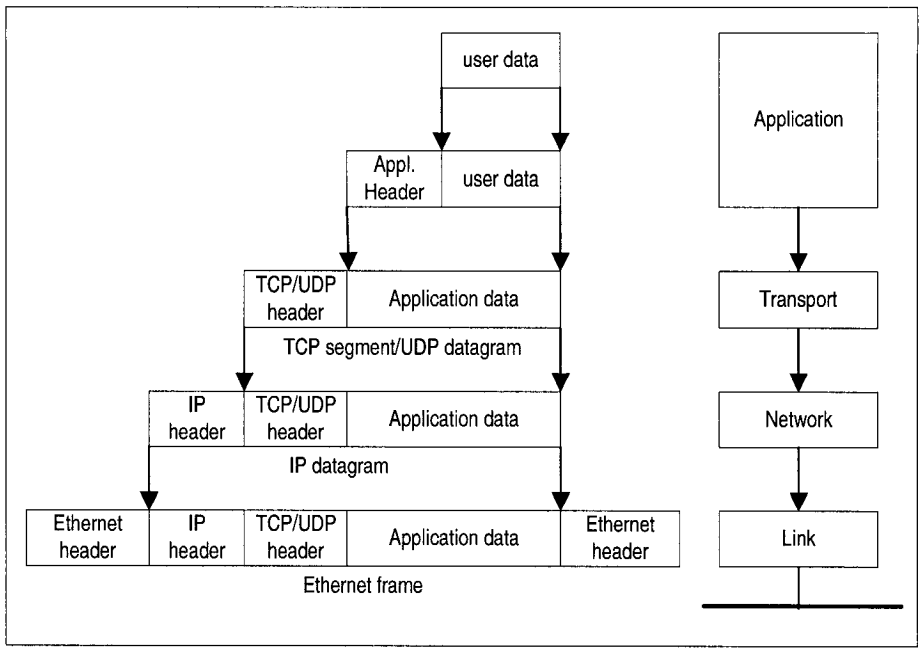

Figure 2-2 Encapsulation and layer distribution

2.2 SDP

On the Internet, as a session directory tool, the Session Description Protocol (SDP) "is used to advertise multimedia conferences and communicate the conference addresses and conference tool-specific information necessary for participation" [RFC2327].

SDP assists in the advertisement of conference sessions and transfer of the relevant conference setup information to prospective participants. In general, SDP must convey sufficient information in order to let participants join a session.

SDP is purely a format for session description. It is independent of a transport protocol, but is designed to cooperate with different protocols as appropriate, such as the Session Initiation Protocol (SIP).

SDP can be used for a wider range of network environments and applications as it is intended to be general purpose. However, it does not support session negotiation (supported by SIP, for example) and media encodings (e.g., by RTP).

SDP has two primary functions: to communicate an existing session, and to transfer sufficient information for establishing a new session. Thus SDP content should include [RFC2327]:

- Session name and purpose
- Time(s) the session is active
- The media comprising the session
- Information to receive those media (addresses, ports, formats and so on)
- Information about the bandwidth to be used by the conference
- Contact information for the person responsible for the session

2.2.1 SDP Specification

"SDP session descriptions are entirely textual using the ISO 10646 character set in UTF-8 encoding. SDP field names and attributes names use only the US-ASCII subset of UTF-8, but textual fields and attribute values may use the full ISO 10646 character set" [RFC2327].

An SDP session description consists of a number of lines of text of the form

<type>=<value>.

<type> is always exactly one character and is case-significant.

<value> is a structured text string whose format depends on <type>. It might be case-significant, depending on a specific definition of the field.

Whitespace is not permitted either side of the '=' sign. In general <value> is either a number of fields delimited by a single space character or a free format string.

An example SDP description cited from [RFC2327] is:

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
e=mjh@isi.edu (Mark Handley)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 31
m=application 32416 udp wb
a=orient:portrait
```

The detailed session description is listed below (Optional items are marked with a '*'.):

```
v=  (protocol version)
o=  (owner/creator and session identifier).
s=  (session name)
i=* (session information)
u=* (URI of description)
e=* (email address)
p=* (phone number)
c=* (connection information - not required if included in all media)
b=* (bandwidth information)
One or more time descriptions (see below)
z=* (time zone adjustments)
k=* (encryption key)
a=* (zero or more session attribute lines)
Zero or more media descriptions (see below)
```

Time description

t= (time the session is active)
r=* (zero or more repeat times)

Media description

m= (media name and transport address)
i=* (media title)
c=* (connection information - optional if included at session-level)
b=* (bandwidth information)
k=* (encryption key)
a=* (zero or more media attribute lines)

Some SDP types that are of special/major importance to this thesis will be discussed a little bit further.

- The origin field o=<username> <session id> <version> <network type> <address type> <address>

  The parameters of the origin field will together form a unique identifier for the current session. The last parameter contains an address, which could be an IPv4 address, an IPv6 address, or a fully qualified domain name.

- The connection field c=<network type> <address type> <connection address>

  The connection field can exist in both the session description and the media description part of the SDP. Its purpose is to give the port number given in the media field an address to be associated with. If this connection field exists in both the session description and the media description part, the connection field in the media description part has a higher priority than the other one.

- The media field m=<media> <port> <transport> <fmt list>

The purpose of the media field is to let the peer party in the session know what kind of media should be delivered to what port on the associated connection address, and how the media should be encoded.

2.3 SIP

The Session Initiation Protocol (SIP) has been described in [RFC3261] by SIP working group of the Internet Engineering Task Force (IETF).

SIP is an application-layer control protocol (see Figure 2-3) that can initiate, modify, and terminate interactive communication sessions between users. Such sessions include voice, video, chat, interactive games, and virtual reality.

SIP establishes sessions by its invitations in which session descriptions are used to negotiate a set of compatible media types to be shared among participants. It can also invite participants to already existing sessions such as multicast conferences. SIP transparently supports name mapping and redirection services, which supports personal mobility. SIP proxy servers could be used to facilitate routing SIP requests to the user's current location. SIP also provides a registration function that stores users' current locations, which could be used by proxy servers to redirect requests.

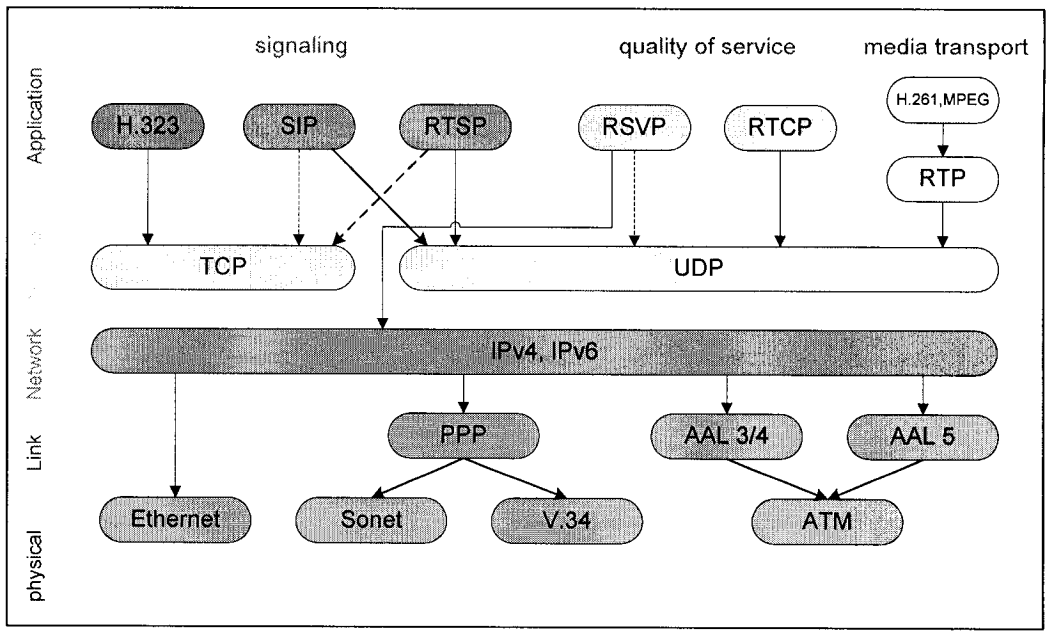

Figure 2-3 Relationship between SIP and other protocols

The characteristics of SIP are simplicity and flexibility. SIP is not a complete communication system. SIP is rather a component that in cooperation with other IETF protocols can provide complete services to the users. Typically, the Real-time Transport Protocol (RTP) [RFC1889] could be used with SIP to support real-time data transfer and provide QoS feedback; the Session Description Protocol (SDP) [RFC2327] could be used for describing multimedia sessions; the Real-Time Streaming Protocol (RTSP) [RFC2326] could be used to control delivery of streaming media; and the Media Gateway Control Protocol (MEGACO) [RFC3015] could be used for controlling gateways to the Public Switched Telephone Network (PSTN).

What should be noted is that SIP does not depend on any of the protocols above that provide services. Rather, SIP provides basic functionalities and operations that can be used to implement different services.

2.3.1 Protocol Specification

2.3.1.1 Protocol Services

SIP supports five facets of establishing and terminating multimedia communications [RFC3261]:

- User location: determination of the end system to be used for communication.
- User availability: determination of the willingness of the called party to engage in communications.
- User capabilities: determination of the media and media parameters to be used;
- Session setup: "ringing", establishment of session parameters at both called and calling party.
- Session management: including transfer and termination of sessions, modifying session parameters, and invoking services.

SIP is not a vertically integrated communications system. SIP is rather a component that can be used with other IETF protocols to build a complete multimedia architecture. In other words, SIP provides primitives that can be used to implement different services.

In addition, SIP provides a suite of security services, which include denial-of-service prevention, authentication (both user to user and proxy to user), integrity protection, and encryption and privacy services.

2.3.1.2 Protocol Vocabulary and Message Format

SIP defines two distinct types of messages: requests and responses. A SIP message is either a request from a client to a server, or a response from a server to a client. Both types of messages consist of a start-line, one or more header fields, an empty line indicating the end of the header fields, and an optional message-body.

```
generic-message  =  start-line
                    *message-header
                    CRLF (Carriage-Return, Line-Feed)
                    [message-body]
start-line       =  Request-Line / Status-Line
```

Every SIP message begins with a Start Line, which may be either a Request-line for requests or a Status-line for responses. The Start Line conveys the message type (method type in requests or response code in responses) and the protocol version.

SIP header fields are used to convey message attributes and to modify message meaning [Radvision01]. Each header field consists of a field name followed by a colon (":") and the field value, i.e., <field-name>:<field-value>. Headers can span multiple lines. Some SIP headers such as Via, Contact, Route and Request-Route can appear multiple times in a message or, alternatively, can take multiple comma-separated values in a single header.

A message body is used to describe the session to be initiated (for example, in a multimedia session this may include audio and video codec types, sampling rates, etc.), or alternatively it may be used to contain opaque textual or binary data of any type that relates in some way to the session. Possible body types include SDP, Multipurpose Internet Mail

Extensions (MIME), and others to be defined in IETF and in the specific implementations [Radvision01].

2.3.1.2.1 Request Messages

SIP defines six SIP request methods as follows:

- INVITE is to initiate sessions. The INVITE method indicates that the user or service is being invited to participate in a session.
- ACK is to confirm session establishment. The ACK request confirms that the client has received a final response to an INVITE request. The ACK request does not generate responses for any transport protocol.
- OPTIONS is to request information about capabilities.
- BYE is to terminate a session. The user agent client uses BYE to indicate to the user agent server that it wishes to release the call leg. A BYE request is forwarded to the server like an INVITE request and may be issued by either caller or callee.
- CANCEL is to cancel a pending session, i.e., the CANCEL request cancels a pending request.
- REGISTER allows a client to bind a permanent SIP Uniform Resource Identifier (URI) to a temporary SIP URI reflecting the current network location. A client uses the REGISTER method to bind the address listed in the To header field with a SIP server to one or more URI where the client can be reached.

2.3.1.2.2 Response Messages

SIP responses are distinguished from requests by having a Status Code in their start-line. The Status-Code is a 3-digit integer result code that indicates the outcome of an attempt to understand and satisfy a request. The first digit of the Status-Code defines the class of response. SIP allows six values for the first digit:

- 1xx: Provisional -- request received, continuing to process the request; e.g., 100 – Trying, 180 – Ringing.
- 2xx: Success -- the action was successfully received, understood, and accepted; e.g., 200 – OK.
- 3xx: Redirection -- further action needs to be taken in order to complete the request; e.g., 302 – Moved Temporarily.
- 4xx: Client Error -- the request contains bad syntax or cannot be fulfilled at this server.
- 5xx: Server Error -- the server failed to fulfill an apparently valid request.
- 6xx: Global Failure -- the request cannot be fulfilled at any server.

2.3.1.3 SIP Entities

A SIP system could consist of many kinds of logical entities, but the essential entities are user agent, proxy server, redirect server, registrar server and location server. They are interconnected as Figure 2-4.

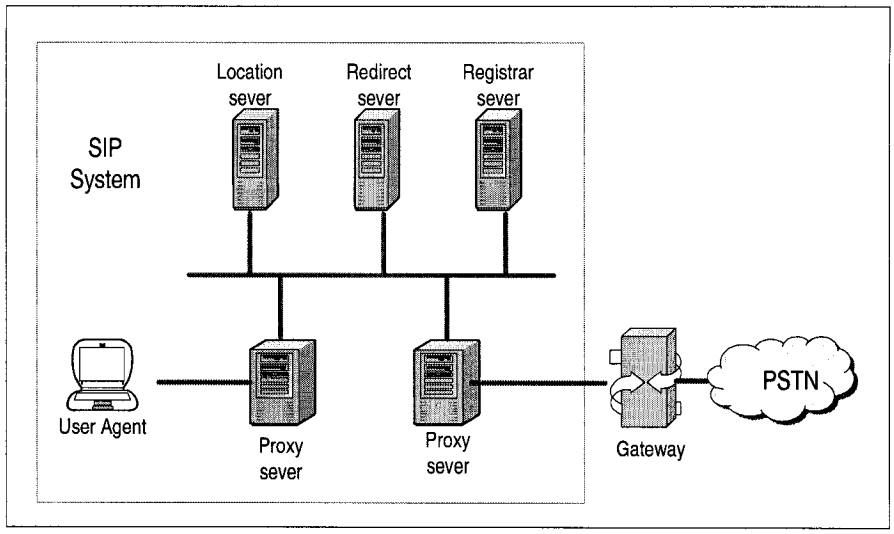

Figure 2-4 SIP Components [Vocal03]

- User Agent (UA): A logical entity that can act as both a user agent client and user agent server. A user agent client (UAC) is a logical entity that creates a new request. A user agent server (UAS) is a logical entity that generates a response to a SIP request.
- Proxy Server: An intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other clients. A proxy server plays the primary role of routing, which means its job is to ensure that a request is sent to another entity "closer" to the targeted user. A proxy interprets, and, if necessary, rewrites specific parts of a request message before forwarding it.
- Redirect Server: A redirect server is a user agent server that generates 3xx responses to requests it receives, directing the client to contact an alternate set of URIs.

- Registrar: A registrar is a server that accepts REGISTER requests and places the information in those requests it receives into the location service for the domain it handles.
- Location Server: A location server is used by a SIP redirect or proxy server to obtain information about a callee's possible location(s).

2.3.1.4 Procedure Rules

SIP requests can be sent directly from a user agent client to a user agent server, or they can traverse one or more proxy servers along the way. User agents send requests either directly to the address indicated in the SIP URI (Uniform Resource Identifier) or to a designated proxy, i.e., outbound proxy, independent of the destination address. The current destination address is carried in the Request-URI. Each proxy can forward the request based on local policy and information contained in the SIP request. The proxy may rewrite the request URI.

A session is initiated with the INVITE request. A successful SIP invitation consists of two requests, INVITE followed by ACK. The INVITE request asks the callee to join a particular conference or establish a two-party conversation. After the callee has agreed to participate in the call with a series of response messages, the caller confirms that it has received those responses by sending an ACK request.

Figure 2-5 shows the basic functions of SIP: location of an end point, signaling of a desire to communicate, negotiation of session parameters to establish the session, and

teardown of the session once established. This scenario shows a typical example of a SIP message exchange between two users, User Agent 1 (UA1) and User Agent 2 (UA2). Each message is labeled with a number.

Assume that all entities have been registered well at the Redirect server functioning as a registrar server. The UA1 initiates a call by sending the request 1 (INVITE message) to its proxy server, Proxy Server 1 (PS1). PS1 has no knowledge about the destination, so it sends the request to a redirect server (RS) to query the location of the remote user agent. The RS returns a 302 response indicating the location, which is a URI of the destination's proxy server (PS2). Then, PS1 rewrites the INVITE message (request 4) and forwards it to PS2 on behalf of UA1. PS2 receives the message and retrieves that its destination is UA2, so it forwards the message to UA2. UA2 retrieves and accepts the SDP information from the INVITE message. It rings the SIP phone and returns a 180 Ringing response to UA1 via the opposite paths (through the response 6, 7 and 8). Thus, UA1 goes to the state of waiting UA2 to take the call. Once UA2 on-hooks, it sends a 200 OK response with its SDP information back to UA1 (through the response 9, 10 and 11). From this point, UA1 is able to directly send its acknowledgement ACK (request 12) to UA2 because the location of the callee is known, and the SIP call has been established over media channel (message 13).

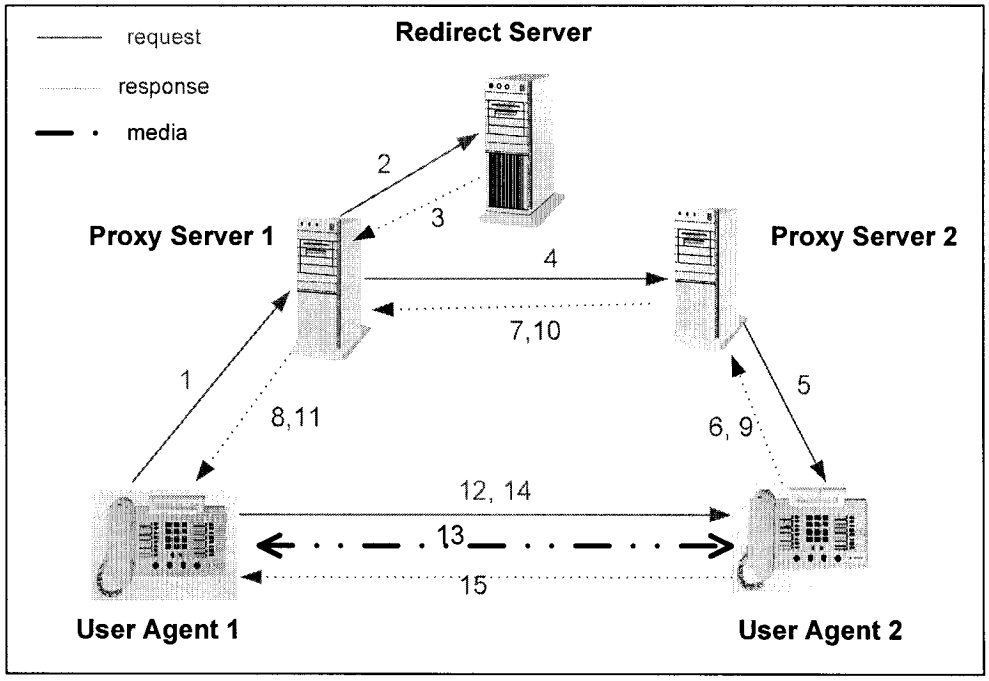

Figure 2-5 Flow Chart of a SIP Call Establishment and Termination

To tear down the SIP call, if UA1 starts a BYE message (request 14) to UA2, UA2 stops the call and responds a 200 OK (response 15) to indicate UA1 that the call has been terminated successfully.

An example of the UA1's INVITE message:

INVITE sip:UA2@ua2-domain.com SIP/2.0
Via: SIP/2.0/UDP ua1-host.ua1-domain.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: UA2 <sip: UA2@ua2-domain.com>
From: UA1 <sip:ua1@ua1-domain.com >;tag=1928301774
Call-ID: a84b4c76e66710@ ua1-host.ua1-domain.com
CSeq: 1 INVITE
Contact: <sip:ua1@ua1-host.ua1-domain.com >
Content-Type: application/sdp
Content-Length: 142

(the SIP body -- UA1's SDP in this case not shown)

RFC 3261 has very detailed descriptions of SIP operations and definitions of SIP messages. This thesis paper does not intend to list all of them. For more information, please refer to [RFC3261].

3 NAT-PT and ALG

3.1 IP Addressing Issues

The Internet Protocol Version 4 (IPv4) is one of the most popular protocols in use today. However, IPv4 is more than 30 years old and has started to show its age. There are some questions about its capability to serve the Internet community any longer. The main issue is the lack of addressing; many experts believe that we are nearly out of the four billion addresses available in IPv4. Although 32-bit addressing of IPv4 seems like a very large number of addresses, multiple large blocks are given to government agencies and large organizations. "There are 4 billion Internet addresses in the world, of which 70 percent have been allocated to North America, leaving the rest of the connected world with only 30 percent. When Stanford University has more IP addresses than the whole of China, you realize the scale of the imbalance" [Rahamim02]. The Internet Protocol Version 6 (IPv6), which uses a 128-bit addressing method replacing the 32-bit addressing of IPv4, could be the solution to many problems.

The timeframe for upgrading the current IPv4 devices to IPv6 is difficult to foresee, thus IPv4 and IPv6 must coexist for a period of time. Connecting IPv6 endpoints to IPv4 endpoints (and vise versa) requires the use of some sort of translation mechanism [Flykt01]. The Network Address Translation - Protocol Translation (NAT-PT) is an IPv4-to-IPv6 transition mechanism aimed at solving such end-to-end IP incompatibilities.

IPv4 and IPv6 have some differences as below:

- They have different header structure:

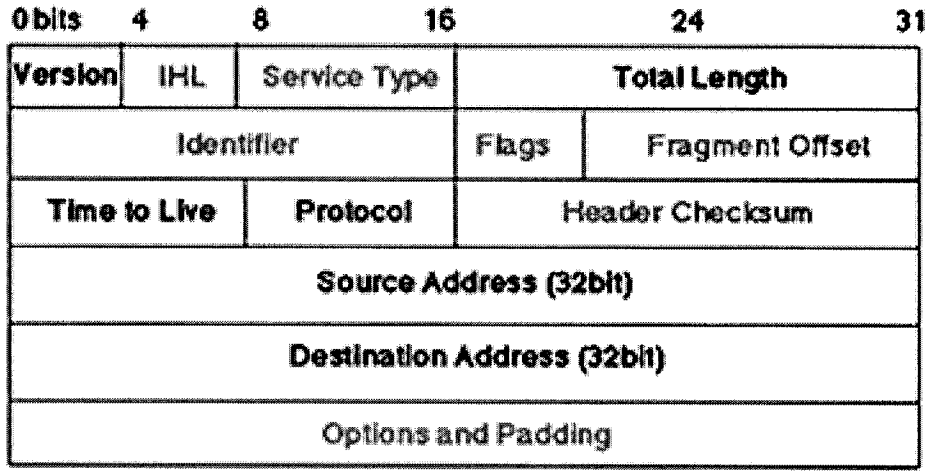

Figure 3-1 IPv4 Header [IPv603]

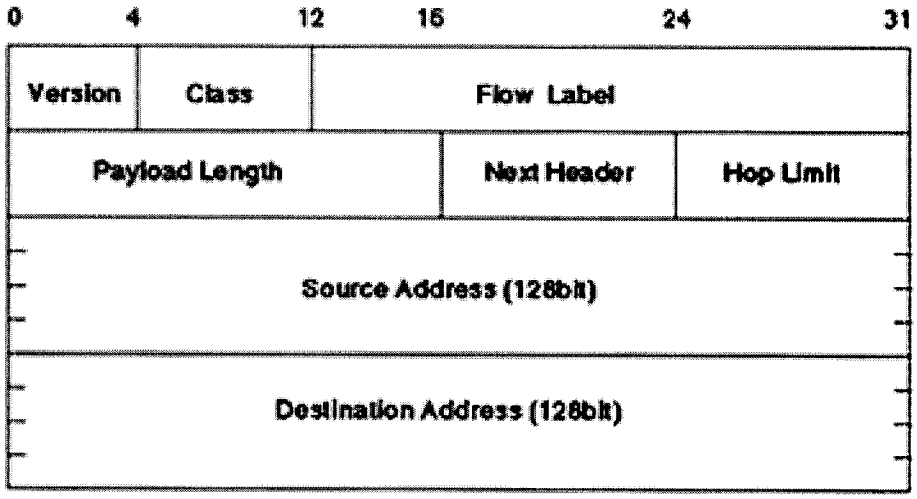

Figure 3-2 IPv6 Header [IPv603]

- Both have different text representations:

IPv4 addresses are 32 bits long, while IPv6 interfaces are identified by 128-bit addresses. Also, IPv4 address uses dotted decimal format, e.g., 142.133.73.99. However, IPv6 address looks like fec0:0:0:1:250:b7ff:fe14:3617. The address is

an eight-part hex address separated by colons (":"). Each part equals a 16-bit number.

3.2 NAT-PT

In principle, NAT-PT can be used to serve both IPv6 and IPv4 stub-domains, offering them connectivity to the IPv4 and the IPv6 network respectively. NAT-PT is based on NAT-like address translation and IP header conversion. It does not have special routing requirements and tunneling support.

When packets for session initiation traverse IPv4/IPv6 boundaries, NAT-PT uses a pool of IPv4 address for assignment to IPv6 nodes on a dynamic basis. The IPv4 addresses in the pool should be globally routable. NAT-PT binds addresses in the IPv6 network to addresses in the IPv4 address pool (and vice versa) to provide transparent routing for the packets moving between the different address domains. This requires no change to endpoints and IP packet routing mechanisms within either IPv6 domain or IPv4 domain. However, it does require NAT-PT to track the sessions it supports and mandates that inbound and outbound packets pertaining to a session traverse the same NAT-PT device. [Bajko01]

A fundamental requirement of the NAT-PT mechanism is that it must be a dual stack component standing on the boundary between the IPv6 stub domain and the IPv4 Internet and further that it must make a link between the two realms.

NAT-PT operates with two sorts of address pools:

- IPv4 Pool

  IPv4 pool contains the global IPv4 address space from which addresses will be allocated dynamically to represent nodes from the IPv6 stub domain. This will typically be a "small" IPv4 address pool identifiable via one IPv4 prefix.

- IPv6 Pool

  IPv6 pool contains the (global) IPv6 address space from which addresses will be allocated dynamically to represent nodes from the IPv4 network. This will typically be a prefix of length 96-bit that can be pre-pended to all IPv4 addresses, which will compose IPv6 addresses being routable in the IPv6 stub domain.

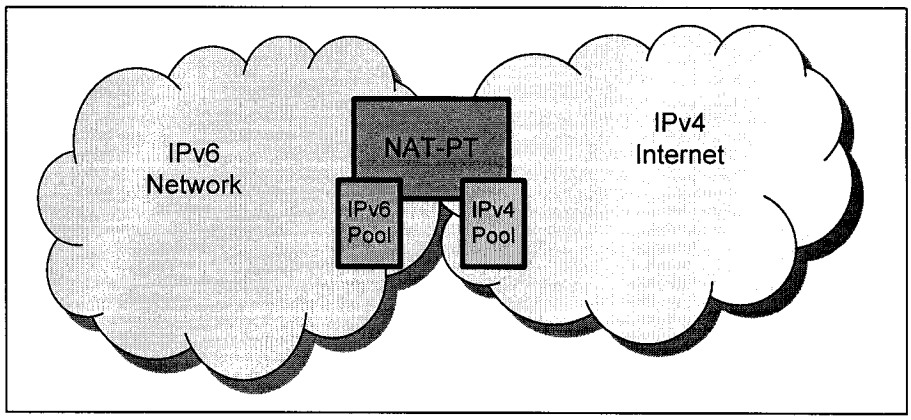

Figure 3-3 NAT-PT Box between IPv4/IPv6 Boundary

3.3 ALG, DNS-ALG

Because NAT-PT is only responsible for translating between the IPv4 and IPv6 headers, but does not snoop the payload of IP packets, NAT-PT requires some Application Level Gateways (ALG) for the specific applications whose packets carry IP addresses in IP

payloads. This said, the most connectivity between the IPv6 stub domain and the IPv4 Internet are achieved by a tight coupling between the NAT-PT functionality and the ALGs.

It is not possible to construct a generic ALG that solves the problem for all possible applications. However, it is possible for application X to solve the problem by implementing X-ALG. X-ALG understands the structure of IP packets sent by the application X and knows which addresses should be changed according to the address mapping database in NAT-PT box [Kristensen02].

Because most applications require the Domain Name System (DNS) and DNS packets carrying the IP addresses in their payloads, a DNS-ALG must be employed in conjunction with NAT-PT to facilitate name-to-address mapping. Specifically, the DNS-ALG must be capable of translating IPv6 addresses in DNS queries and responses into their IPv4 address bindings allocated by NAT-PT, and vice versa, as the DNS packets traverse the IPv4/IPv6 boundaries.

I do not intend to introduce the detailed information about the DNS-ALG—being outside the scope of this thesis—but rather give some basic operations of DNS-ALG.

3.3.1 DNS-ALG and NAT-PT basic operations

In the DNS packets, A records represent IPv4 addresses, while AAAA or A6 records represent IPv6 addresses.

3.3.1.1 DNS Query Originating from IPv6 Side to IPv4 Side

1. All IPv6 name queries must go through the DNS-ALG directly. The DNS-ALG intercepts the DNS requests for AAAA DNS records sent from the IPv6 endpoints and acts as a "proxy-resolver" by forwarding the requests for both A and AAAA records to the DNS system (IPv4 DNS servers and IPv6 DNS servers respectively).
2. Upon receiving the answers from the DNS system, the DNS-ALG returns AAAA records to the inquiring IPv6 hosts by one of the following ways:
   - In case AAAA record is found, this record is returned to the IPv6 host and all resolved A records are discarded.
   - In case such that only A record is returned, the DNS-ALG allocates an IPv6 address from the IPv6 pool in NAT-PT for the resolved IPv4 host. Typically, DNS-ALG and NAT-PT append the IPv4 address (A record) to a 96-bit prefix and respond it to the inquiring IPv6 host as an AAAA record.

     (Note: NAT-PT will create an address mapping for the IPv6 host and its corresponding IPv4 address allocated from IPv4 address pool.)
3. The session sent from the IPv6 host to the IPv4 host (with the allocated IPv6 address) will at this time be intercepted by the NAT-PT module, which will translate the IPv6 packets into IPv4 ones by the corresponding address mapping. Similarly, in the opposite direction, the communication sent from IPv4 host to the IPv6 host (with the mapped IPv4 address) will be modified by the NAT-PT correspondingly as well.

3.3.1.2 DNS Query Originating from IPv4 Side to IPv6 Side

1. All IPv4 DNS queries will be sent to local DNS system in IPv4 side. If no answers are found, the queries will be directed to the IPv6 DNS server through NAT-PT box in which DNS-ALG will intercept the queries.
2. Upon receiving the DNS query, DNS-ALG translates A record to AAAA record before forwarding it to IPv6 DNS server in the IPv6 stub domain.
3. When an IPv6 DNS response is received from the IPv6 DNS server for the inquiring IPv4 host, the DNS-ALG once again intercepts the DNS packet, and translates it from AAAA record to A record. Importantly, DNS-ALG requires NAT-PT to allocate an IPv4 address from the IPv4 address pool for the queried IPv6 address. Also, NAT-PT will create a mapping for this pair of IPv4 and IPv6 addresses.
4. Similar to the third step of the Section 3.3.1.1, NAT-PT box will intercept and modify packets according to the address mapping during the communication between the IPv4 host and the IPv6 host.

4 SIP-ALG

There exists an increasing need for IP address translation because the networks based on IPv6 addresses have being extended and because a network's private IP addresses cannot be used outside the network. The NAT/NAT-PT devices could be utilized to satisfy the demand; however, different Application Level Gateways (ALG) are still needed for different network applications once they make use of network address in their message bodies.

SIP is an application layer control protocol for establishing media sessions. It encounters problems with NAT-like devices, because the payloads of SIP packets carry the addresses for the sessions to be established. However, NAT-PT is unaware of application and does not snoop into the payloads. With NAT-PT and DNS-ALG, a SIP-ALG is needed for SIP sessions on the boundary between IPv4 and IPv6.

This Chapter describes an implementation of a SIP-ALG to enable typical SIP sessions to pass through a NAT-PT box based on the Vocal SIP user agent application (an open source of SIP implementation created by Vovida.org [Vocal03]). Rather than attempt to make a full specification for SIP-ALG, I have implemented a subset of the functionalities that is sufficient for typical use.

4.1 SIP header fields

Before describing the behavior of a SIP-ALG, it is necessary to understand what content of a SIP message should be modified by the ALG. From the Section 2.3.1.2, we know that a SIP message consists of a start-line, numerous header fields, and an optional message body. Below is a typical SIP packet generated by the Vocal system.

```
INVITE sip:ying@20.0.0.2:5060 SIP/2.0
Via: SIP/2.0/UDP 20.0.0.1:5060
To: <sip: ying@20.0.0.2:5060>
From: UA-aloha<sip:aloha@20.0.0.1:5060;user=phone>
Call-ID: 8de9e4d65d67f870c34e85705792765e@20.0.0.1
CSeq: 1 INVITE
Max-Forwards: 70
Subject: VovidaINVITE
Contact: <sip:aloha@20.0.0.1:5060;user=phone>
Content-Type: application/sdp
Content-Length: 214

v=0
o=- 342351072 342351072 IN IP4 20.0.0.1
s=VOVIDA Session
c=IN IP4 20.0.0.1
t=3259839490 0
m=audio 10100 RTP/AVP 0 100
a=rtpmap:0 PCMU/8000
a=rtpmap:100 telephone-event/8000
a=ptime:20
a=fmtp:100 0-11
```

Figure 4-1 A Typical SIP Request Message

SIP is a text-based protocol. The first line of the message, the start-line, may contain the address of the next hop that could be either the next SIP server or the

destination. The lines following the start-line are a list of header fields and message body. I only introduce some of those related to the SIP-ALG.

- "Via" contains the address at which the caller (aloha) is expecting to receive responses to its requests. It will be added/removed across SIP proxy servers that route SIP messages.
- "To" contains the address of the original destination (ying) of the SIP call that is generated from DNS lookup.
- "From" contains the address that indicates the originator of the request.
- "Contact" contains the address that represents a direct route to tell other hosts where to send future messages. For SIP requests, Contact is the address of the caller; for SIP responses, Contact is the address of the callee.
- "Content-Type" is 'application/sdp', which implies the message body, SDP message in this example, must be modified as well.
- "Content-Length" contains an octet count of the message body that should be modified correspondingly after address conversion.
- The SDP message starts with a 'v=' line. Two lines should be taken account into the modification. The 'o=' line contains the owner of the SDP message. The 'c=' line contains the connection information. Further information about these two lines is given in Section 2.2.1.

4.2 SIP-ALG behavior

An IP packet carrying a SIP message is identified by NAT-PT box by the characteristic of the SIP protocol, using the port 5060 as the destination.

Whenever a Vocal user agent (UA) initiates a SIP session by the host name of a callee, it looks up the IP address from DNS services. DNS servers transparently provide the address as usual except that the NAT-PT will set up an address mapping once the DNS query has traversed the boundary between IPv4 and IPv6, which has been described in Section 3.3.1. If a mapping existed, the SIP message is sent to the NAT-PT box because the current destination address will be an address from IPv4/IPv6 address pool directed (routed) to the NAT-PT machine. The SIP-ALG in the NAT-PT box will build a table storing two pairs of the source address and the corresponding destination address for both IPv4 and IPv6 domains. According to the table, the start-line and the Via, To, From, Contact field in the SIP message will be modified. If the Content-Type is SDP, the SDP message and the Content-Length will also be adjusted. After IP header is also converted between IPv4 and IPv6 by NAT-PT module, the modified message is forwarded to another IP version of network. Similarly, the response messages will be modified back to the original messages appropriately when they return to the NAT-PT box. Thus, it seems that the IPv4 UA and the IPv6 UA are communicating directly, but actually communicating with the NAT-PT box respectively.

I will describe the detailed behavior using an example. Figure 4-2 shows a simple network setup.

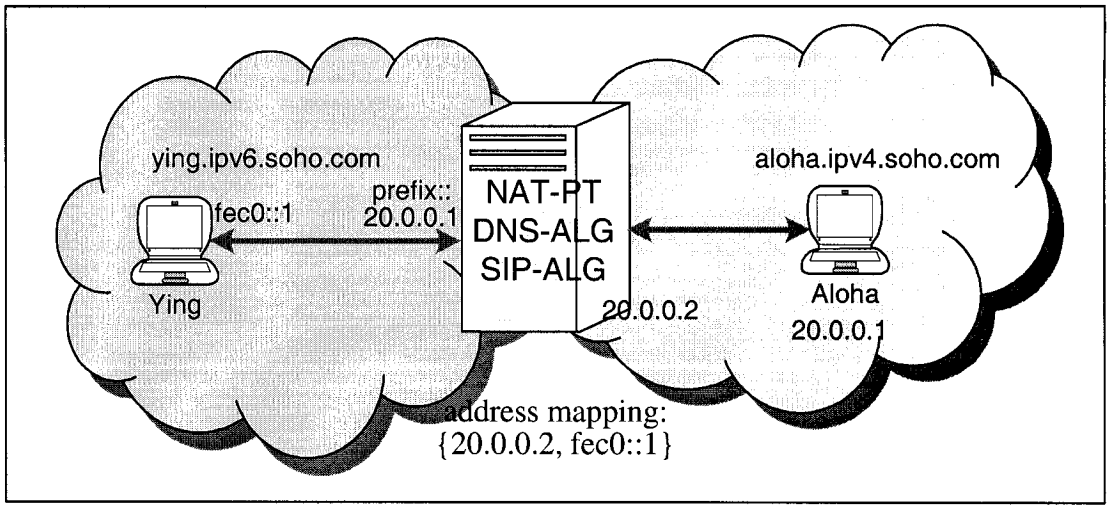

Figure 4-2 Network Setup with NAT-PT box

Ying is a UA in IPv6 network with IPv6 address fec0::1. Aloha is a UA in IPv4 network with IPv4 address 20.0.0.1.

Assume that aloha starts a SIP call to ying by ying's host name (ying.ipv6.soho.com).

Aloha's UA tries to get the IP address for the host name. When the result of the DNS query (fec0::1) is returned to DNS-ALG, it creates an address mapping between fec0::1 and an allocated address (20.0.0.2 in this example) from the IPv4 address pool.

The DNS-ALG sends the result (20.0.0.2) to aloha's UA. The UA creates a SIP INVITE message (see Figure 4-1) and sends to 20.0.0.2 (NAT-PT box) with the port 5060.

The NAT-PT box captures the IPv4 packet at the port 5060. It retrieves the IP header and transforms the destination address to fec0::1 replacing 20.0.0.2.

The NAT-PT box then sends the IP payload, i.e., UDP packet, to the SIP-ALG. The SIP-ALG generates a table for SIP address translation. In this example, the table is:

SIP_v4_source: 20.0.0.1 SIP_v4_dest: 20.0.0.2

SIP_v6_source: prefix:20.0.0.1 SIP_v6_dest: fec0::1

The SIP-ALG retrieves the SIP message from the UDP packet, and takes the following actions:

1. Replace 20.0.0.1 with prefix:20.0.0.1 for the From, Via, and Contact fields and 'o=' and 'c=' lines.
2. Change 'IN4' to 'IN6' in the two lines.
3. Replace 20.0.0.2 with fec0::1 for the start-line and To field.
4. Calculate the octet count of the new SDP message and modify the Content-Length field.

In addition, the SIP-ALG calculates the length of the SIP message and updates the payload length of the UDP packet that will be used to upgrade the IP header at the NAT-PT module. Figure 4-3 illustrates a main part of the INVITE message modified by the SIP-ALG corresponding to the message shown in Figure 4-1. The value of Content-Length is set to "xxx" because the real value has not been calculated here.

Note that each IPv6 address in this implementation is enclosed by a pair of "[" and "]" in order to distinguish IPv6 addresses and IPv4 addresses.

INVITE sip:ying@[fec0::1]:5060 SIP/2.0
Via: SIP/2.0/UDP [prefix:20.0.0.1]:5060
To: <sip: ying@[fec0::1]:5060>

From: UA-aloha<sip:aloha@[prefix:20.0.0.1]:5060;user=phone>
... ...
Contact: <sip:aloha@20.0.0.1:5060;user=phone>
Content-Length: xxx
... ...
o=- 342351072 342351072 IN IP6 [prefix:20.0.0.1]
c=IN IP6 [prefix:20.0.0.1]

Figure 4-3 SIP Request Modified by SIP-ALG (IPv4 to IPv6)

The NAT-PT module reassembles and sends the IPv6 packet to the domain ipv6.soho.com.

In the Vocal implementation, ying's UA will keep the address part of Via, To and From fields in its responses, but change the address of Contact field and SDP message in 200 (OK) response to its IPv6 address. It does not contain any network address in the start-line.

SIP/2.0 200 OK
Via: SIP/2.0/UDP [prefix:20.0.0.1]:5060
To: <sip: ying@[fec0::1]:5060>
From: UA-aloha<sip:aloha@[prefix:20.0.0.1]:5060;user=phone>
... ...
Contact: <sip: ying@[fec0::1]:5060;user=phone>
Content-Length: xxx
... ...
o=- 342351072 342351072 IN IP6 [fec0::1]
c=IN IP6 [fec0::1]

Figure 4-4 SIP Response (IPv6 to IPv4)

When the NAT-PT box receives the response from ying, it transforms the IP header and sends the IP payload to the SIP-ALG.

Again, the SIP-ALG builds a table:

SIP_v6_source: fec0::1 SIP_v6_dest: prefix:20.0.0.1

SIP_v4_source: 20.0.0.2 SIP_v4_dest: 20.0.0.1

Similarly, for the Via, To, From, Contact fields and SDP message, replace fec0::1 and prefix:20.0.0.1 to 20.0.0.2 and 20.0.0.1 respectively. Also update the Content-Length and payload length of the UDP packet correspondingly.

SIP/2.0 200 OK
Via: SIP/2.0/UDP 20.0.0.1:5060
To: <sip: ying@20.0.0.2:5060>
From: UA-aloha<sip:aloha@20.0.0.1:5060;user=phone>
... ...
Contact: <sip:ying@20.0.0.2:5060;user=phone>
Content-Length: xxx
... ...
o=- 342351072 342351072 IN IP4 20.0.0.2
c=IN IP4 20.0.0.2

Figure 4-5 SIP Response Modified by SIP-ALG (IPv6 to IPv4)

Thus, the IPv4 UA and the IPv6 UA communicate transparently across the boundary between the IPv4 and IPv6 networks.

5 Original SIP Security Mechanism

SIP deployments in a VoIP (Voice over IP) network are exposed to a wide range of network security threats and attacks. SIP, however, is not an easy protocol to secure because of its use of intermediaries, its multi-faceted trust relationships, its expected usage between elements with no trust, and its user-to-user operation. All of this makes security a real issue. [RFC3261]

In this chapter, I would like to overview some classic threat models first, then enumerate the security mechanism that SIP has supported according to its Request For Comments [RFC3261].

5.1 Threat Models

This section introduces some attacks that are common to most deployments of SIP.

These attacks assume an unsecured environment such as the public Internet in which attackers can potentially read any packet on the network. Attackers on the network may be able to modify packets, steal services, eavesdrop on communications, or disrupt sessions (perhaps at some compromised intermediary).

5.1.1 Registration Hijacking

Registration creates bindings in a location service for a particular domain. These address bindings map an incoming SIP or SIPS URI (associated with the To header field)

to one or more URIs (associated with the Contact header field) that are somehow "closer" to the desired user.

In a REGISTER request, the typical header fields represent:

- Contact: REGISTER requests may contain a Contact header field with zero or more values containing address bindings.
- Request-URI: The Request-URI names the domain of the location service for which the registration is meant.
- To: The To header field contains the address-of-record whose registration is to be created, queried, or modified.
- From: The From header field contains the address-of-record of the person responsible for the registration. The value is the same as the To header field unless the request is a third-party registration.

A registrar assesses the identity asserted in the From header field of a REGISTER message to determine whether this request can modify the contact addresses associated with the address-of-record in the To header field. However, the From header field can be modified for a third-party by the owner of a UA. This implies that a door to malicious attack is opened. For example, an attacker can successfully impersonate a party authorized to change contacts associated with an address-of-record; therefore, he could direct all requests for the affected user to his device.

This threat needs the security services that enable SIP entities to authenticate the originators of requests. [RFC3261]

5.1.2 Impersonating a Server

"There is always a possibility that an attacker could impersonate the remote server, and that the UA's request could be intercepted by some other party" [RFC3261].

For example, an attacker impersonates a redirect server at a domain. A user agent sends a request to the domain, but the attacker intercepts and answers the request with a forged response that contains appropriate SIP header fields. The forged Contact addresses in the redirection response could direct the originating UA to inappropriate or insecure resources, or simply prevent requests from succeeding.

To prevent this kind of threat, UAs must be required to authenticate the servers to whom they send requests.

5.1.3 Tampering with Message Bodies

Although SIP UAs route requests via trusted proxy servers, the server may inspect and/or perhaps modify the bodies contained in the requests.

For instance, a UA is using SDP to transmit session encryption keys for a media session. Although it trusts the proxy server of the domain that it is contacting, the administrators of that domain may be able to capture the encryption keys and decrypt any subsequent media session. It is worse that the proxy server acts maliciously. It could modify the session key, either acting as a man-in-the-middle attacker, or lowering the security level requested by the originating UA. As a man-in-the-middle attacker, the

server could intercept messages in a public key exchange and then retransmit them, substituting their own public key for the requested one, so that the two original parties still appear to be communicating with each other directly.

Another example: attackers may attempt to modify SDP bodies to point RTP media streams to a wiretapping device in order to eavesdrop on subsequent voice communications.

To prevent these problems, UAs have to secure SIP message bodies and some header fields if necessary in some end-to-end cases. "The security services required for bodies include confidentiality, integrity and authentication. These end-to-end services should be independent of the means used to secure interactions with intermediaries such as proxy servers" [RFC3261].

5.1.4 Tearing Down Sessions

If a third-party attacker captures some initial messages in a session in order to learn the parameters of the session, such as the To and From headers, a BYE or re-INVITE request can perhaps be inserted into the session. Once the BYE is received by its destination, the session will be torn down prematurely. Upon the re-INVITE, the attacker might reduce the session security or redirect media streams to a wiretapping device.

The most effective protection against this threat is to authenticate or verify that the BYE or re-INVITE came from the same party with whom the corresponding session was

established. In addition, the attacker would be unable to learn the parameters of the session due to securing those parameters as the session is established.

5.1.5 Denial of Service

On the Internet, a distributed denial-of-service (DoS) attack uses a multitude of compromised systems to attack a single target, thereby causing denial of service for users of the targeted system. The flood of incoming messages to the target system essentially forces it to shut down, thereby denying service to the system to legitimate users.

In SIP systems, SIP proxy servers face the public Internet in order to accept requests from worldwide IP endpoints. SIP creates a number of potential opportunities for distributed denial-of-service attacks.

Attackers can falsify the requests directed to a targeted host as the originator of the requests and send the requests to a number of SIP network elements. Afterwards, together these actions will generate denial-of-service traffic aimed to the target.

If REGISTER requests are not properly authenticated and authorized by registrars, attackers can de-register some or all users disabling users to be invited to a new session. Also, attackers can register a huge number of contacts for the same host to amplify the denial-of-service attack and attempt to consume available resources, such as memory and disk space, of a registrar.

Solving these problems requires the design of architectures that minimize the risks of denial-of-service.

5.2 SIP Security Mechanisms

"The fundamental security services required for the SIP protocol are: preserving the confidentiality and integrity of messaging, preventing replay attacks or message spoofing, providing for the authentication and privacy of the participants in a session, and preventing denial-of-service attacks. Bodies within SIP messages separately require the security services of confidentiality, integrity, and authentication" [RFC3261].

Rather than defining new security mechanisms specific to SIP, SIP reuses, wherever possible, existing security models derived from the Hypertext Transfer Protocol (HTTP) and Simple Mail Transfer Protocol (SMTP) space.

The security of SIP signaling itself has no bearing on the security of protocols used in concert with SIP such as RTP, or with the security implications of any specific bodies, e.g., MIME.

Note that the RTP security mechanism is not under the scope of this thesis.

5.2.1 Transport and Network Layer Security

SIP supports the transport or network layer security, such as TLS [RFC2246] or IPSec [RFC2401] respectively, that encrypt signaling traffic and guarantee message confidentiality and integrity.

The Transport Layer Security (TLS) protocol specifies a transport-layer security mechanism over connection-oriented protocols in order to provide communication

privacy and data integrity. The protocol is composed of two layers: the TLS Record Protocol and the TLS Handshake Protocol. The TLS Record Protocol is used for encapsulation of various higher-level protocols. The TLS Handshake Protocol allows the server and client to authenticate each other and negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first packet.

TLS is most suited to architectures in which hop-by-hop security is required between hosts with no pre-existing trust association. There is no assurance that TLS will be used end-to-end as a UA sends requests over TLS to its destination.

The Internet Protocol Security (IPSec) is a network layer protocol suite that provides privacy and authentication services at the network layer. It provides a set of security algorithms and a general framework that allow a pair of SIP entities to secure the signaling traffic. It consists of three protocols: Authentication Header (AH), Encapsulation Security Payload (ESP) and Internet Key Exchange (IKE). The three protocols interoperate with the Security Association (SA). IPSec provides assurance for confidentiality, connectionless integrity, data origin authentication, replay protection and traffic analysis protection. [RFC2401]

- Authentication Header (AH) provides integrity and authenticity of the data, including the invariant fields in the outer IP header. It does not provide confidentiality protection. [RFC2402]

- Encapsulation Security Payload (ESP) provides confidentiality, integrity and authenticity of the data. If ESP is used to validate data integrity, it does not include the invariant fields in the IP header. [RFC2406]
- Internet Key Exchange (IKE) is a key agreement protocol that provides a powerful and flexible negotiation on authentication algorithms, encryption algorithms, the keys to use, and the keys lifetime using a special message called Security Association (SA). [RFC2409]
- Security Association (SA) is a one-way directional agreement between two communicating parties that specify a set of policies and keys to protect the future communication between them. The concept of SA is fundamental to IPSec. Both AH and ESP make use of SAs, and a major function of IKE is the establishment and maintenance of SAs. [RFC2401]

IPSec is most commonly used in architectures in which there is an existing trust relationship among a set of hosts and domains. IPSec is usually implemented at the operating system level in a host or on a security gateway that provides confidentiality and integrity for all traffic it receives from a particular interface. UAs that have a pre-shared keying relationship with their first-hop proxy server are also good candidates to use IPSec. [RFC3261]

5.2.2 SIPS URI Scheme

SIP also provides a secure URI, called a SIPS URI. The format for a SIPS is the same as that of a SIP, except that the scheme is "sips" instead of "sip". An example would be sips:aloha@soho.com. A call made to a SIPS URI guarantees that secure, encrypted transport (namely TLS) is used to carry all SIP messages from the caller to the domain of the callee.

There are two ways to use SIPS:

- When used by the originator of a request (as would be the case if they employed a SIPS URI as the address-of-record of the target), SIPS dictates that the entire request path to the target domain be secured.
- When used as the Request-URI of a request, the SIPS scheme signifies that each hop over which the request is forwarded, until the request reaches the SIP entity responsible for the domain portion of the Request-URI, must be secured with TLS; once it reaches the domain in question it is handled in accordance with local security and routing policy, quite possibly using TLS for any last hop to a UAS.

Note that in the SIPS URI scheme, transport is independent of TLS, but UDP is not a valid transport for SIPS.

5.2.3 HTTP Authentication

SIP provides a stateless, challenge-based mechanism for authentication that is based on authentication in HTTP [RFC2617]. A proxy server or UA may challenge the

initiator of the request to provide assurance of its identity whenever it receives a request. Once the originator has been identified, the recipient of the request should ascertain whether or not this user is authorized to make the request in question.

The framework for SIP authentication closely parallels that of HTTP. In SIP, a UAS uses the response 401 (Unauthorized) to challenge the identity of a UAC. Additionally, registrars and redirect servers may make use of 401 (Unauthorized) responses for authentication, but proxies must not, but instead may use the 407 (Proxy Authentication Required) response.

The "Digest" authentication mechanism provides message authentication and replay protection only, without message integrity or confidentiality.

5.2.4 S/MIME

SIP messages carry MIME (Multipurpose Internet Mail Extension) bodies and the MIME standard includes mechanisms for securing MIME contents to ensure end-to-end integrity and confidentiality, as well as mutual authentication [RFC1847, RFC2630, RFC2633]. It is also possible to use Secure Multipurpose Internet Mail Extension (S/MIME) to provide a form of integrity and confidentiality for SIP header fields through SIP message tunneling.

I do not intend to describe in more detail how to sign and encrypt MIME messages in this thesis paper. For more information, please refer to Section 23 in [RFC3261]. Now I

shall introduce the S/MIME certificates and key exchange, which will turn out to be a weak point of the S/MIME discussed later.

5.2.4.1 S/MIME Certificates

The S/MIME certificates are used to identify an end-user. The certificates assert that the holder is identified by an end-user's address, most often corresponding to the user's address-of-record as the form "userinfo@domainname".

These certificates could be associated with the keys to sign or encrypt SIP messages. The public keys are used to encrypt MIME bodies for the intended recipient, and the private keys of the sender to sign the bodies.

Public keys can be stored within a UA on a virtual keyring. This keyring maps between UA's address-of-record and the corresponding certificates. Whenever the UA generate a request, it should use the same address-of-record and the same corresponding certificate.

There are several ways to obtain certificates. Users should acquire certificates from a few well-known centralized directories. UAs should support a mechanism for publishing and importing certificates in any public directories as appropriate. Another method is that users may create self-signed certificates. Also, administrators may use pre-configured certificates if a previous trust relationship exists among all SIP entities.

5.2.4.2 S/MIME Key Exchange

This section is adapted from [RFC3261].

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. CMS [RFC2630] stands for Cryptographic Message Syntax, which is used to digitally sign, digest, authenticate, or encrypt arbitrary messages. The SignedData, which is one of the content types defined by CMS, consists of a content of any type and zero or more signature values. The SignedData content type is typically applied to represent one signer's digital signature on content of the data content type.

When a UAC sends a request containing an S/MIME body, the UAC should structure the body as an S/MIME 'multipart/signed' CMS SignedData body.

When a UAS receives a request containing an S/MIME CMS body with a certificate, the UAS should first validate the certificate, if possible, with any available root certificates for certificate authorities. It also should determine whether the subject of the certificate matches the From header field of the request. If the certificate cannot be verified, because it is self-signed, or signed by unknown 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 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 for the first request, the UAS should structure the body as an S/MIME 'multipart/signed' CMS SignedData body.

When a UAC receives the response containing an S/MIME CMS body from the UAS, the UAC should first validate the certificate, as the UAS did, but compare the subject of the certificate with the To field of the response. 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, during the course of a session, a UA receives a certificate in a CMS SignedData message that does not correspond with the certificates previously exchanged, the UA must notify its user of the change, preferably in terms that indicate that this is a potential security breach. If the user authorizes this certificate, it should be added to the keyring alongside any previous value(s) for this address-of-record.

6 SIP End-to-end Security with NAT-PT

This chapter will describe a SIP end-to-end security proposal for communication between IPv4 endpoints and IPv6 endpoints based on an existing end-to-end security solution for IPv4-IPv4 communication.

6.1 Security Issues with NAT-PT

SIP messages contain private, sensitive information that should be kept hidden by end users. To achieve this, appropriate security mechanisms must be recruited. Hop-by-hop security mechanisms put SIP proxies in potential un-trusted domains between the endpoints in which session information might be altered or just eavesdropped. The best way to prevent this is to use end-to-end security. Secure Multipurpose Internet Mail Extension (S/MIME) allows the users to encrypt SIP messages end-to-end. After encryption of SIP messages, no intermediaries can inspect the session data. That is, S/MIME may prevent the intermediaries that rely on viewing or modifying the SIP messages from functioning.

On the other hand, as discussed above, IP address incompatibilities between IPv4 addresses and IPv6 addresses require IP address translation for the SIP messages as they are traversing the boundary of IPv4/IPv6 domains in order that the messages can be recognized and routed in each domain properly. In this case, a NAT-PT box including a SIP-ALG is needed to perform this kind of address translation. For the SIP-ALG,

analyzing and modifying SIP messages is mandatory if the messages will be transmitted between different IP address realms.

These two requirements, end-to-end security and IP address translation, seem to contradict each other. However, K. Umschaden, et. al. in their Internet draft [Umschaden03] contribute an end-to-end SIP security with firewall/Network Address Translation (NAT) between private IPv4 addresses and global IPv4 addresses. Based on this, I propose a security architecture for end-to-end SIP message security associating with a NAT-PT box that converts IPv4 and IPv6 addresses.

Umschaden has proposed a solution that enables end-to-end security of SDP together with firewall/NAT traversal. According to his proposal, the end user authorizes a proxy server to encrypt SDP on behalf of the user. Figure 6-1 illustrates a basic architecture for his solution. The proxy determines the capabilities of the receiving party and encrypts the SDP for a peer SIP proxy server in the receiving domain. Using the Middlebox Communications protocol (MIDCOM), each proxy is able to dynamically request NAT bindings for address translation between an IPv4 private address and an IPv4 public address, or control its firewall to open/close pinholes for media flows. I have validated this protocol using modeling techniques in the next chapter.

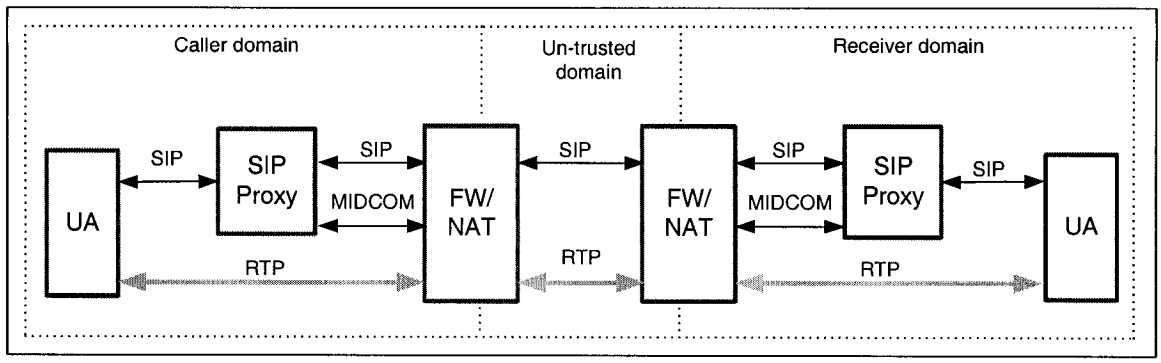

Figure 6-1 Basic Architecture (NAT) [Umschaden03]

Based on Umschaden's solution, it is natural to consider it as a solution to the end-to-end SIP security over the IPv4/IPv6 transition. In Figure 6-1, the SIP proxy is responsible for translating the private IPv4 address to the corresponding global one, and vice versa. The proxy obtains the address binding by using MIDCOM to query the information from the NAT module. That is, the NAT module is transparent to provide the address binding service for the SIP proxy. Hence, it is possible to replace the NAT module with a NAT-PT module with few or no modifications in the SIP proxy. Figure 6-2 shows an architecture for an end-to-end SIP security mechanism with a NAT-PT module and probably with DNS-ALG using MIDCOM protocols.

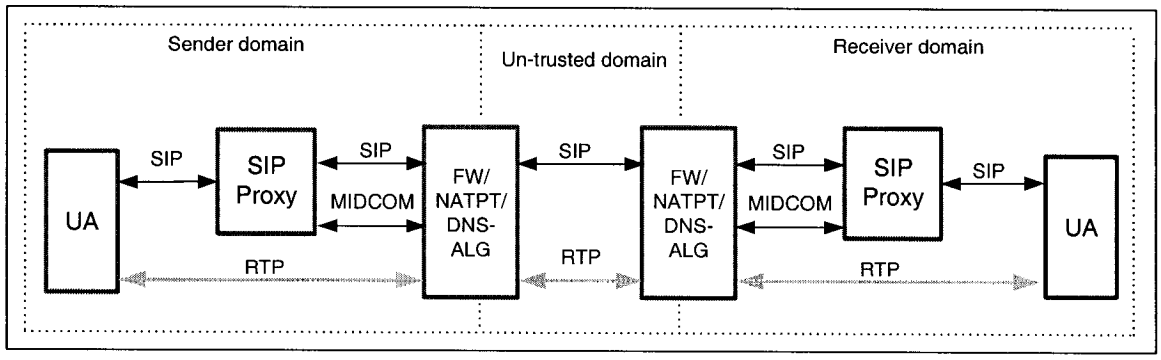

Figure 6-2 End-to-end Security Architecture (NAT-PT/MIDCOM)

Similar to Umschaden's solution, I enable the end-to-end security for communication between IPv4 endpoints and IPv6 endpoints by using the proxy server and the SIP protocol

extension Umschaden made. Unlike his recommendation, a SIP-ALG described in Chapter 4 is used to enable address translation between an IPv4 public address and an IPv6 address for two reasons. The final purpose behind this thesis is to implement the end-to-end SIP security mechanism; however, MIDCOM is still in progress and there is no operable MIDCOM protocol(s), especially no protocols discussing MIDCOM security, just descriptions about its architecture. Another reason is that I have already had a working SIP-ALG, which can facilitate the future implementations. The SIP-ALG is co-located with the NAT-PT modules, so MIDCOM is not needed in this case, and the remote invocation (using MIDCOM) will be changed to the local procedure call originated from the SIP security proxy. Using the SIP-ALG, the system architecture proposed in Umschaden's solution will be changed appropriately, and the signaling negotiation between two end users will perhaps become simpler, which is less time-consuming during the SIP call establishment.

The detailed solution of SIP end-to-end security with a NAT-PT box will be discussed in the following sections.

6.2 Design Goals

1. End-to-end SIP message security should be based on S/MIME. Only secured SIP messages can be transmitted in the un-trusted networks;
2. IP address translation is performed by NAT-PT, SIP-ALG and DNS-ALG modules;

3. Encryption for SIP bodies (SDP message) should not affect the functionality of SIP-ALG.

6.3 Solution – SIP Security Proxy

This section describes a logical SIP entity, namely the security proxy, that enables end-to-end security for SDP together with NAT-PT traversal.

6.3.1 Overview

Ideas for solutions to this end-to-end security mechanism are listed below:

1. Security proxies use S/MIME to encrypt SDP, and send the encrypted messages on the un-trusted networks until the messages arrive at the security proxies in the receiving domain.
2. A security proxy in the sending domain will determine whether the other security proxy in the receiving domain has the capability of supporting this extension mechanism;
3. Establish a secured one-hop path between an end user and a security proxy in the same domain by one of the SIP security mechanism, say TLS;
4. The user is responsible for protecting some SIP header fields by the tunneling technique of S/MIME for privacy reasons;
5. The user authorizes its security proxy to encrypt SIP bodies on its behalf;

6. At least one security proxy is co-located with a NAT-PT box including DNS-ALG and SIP-ALG on the IPv4/IPv6 boundary. The security proxy will encrypt SDP after IP addresses translation by the SIP-ALG for outbound messages; it will decrypt SDP in the received inbound messages before the addresses translation performed by the SIP-ALG.

6.3.2 Syntax

Considering the ideas expressed above, it is necessary to introduce a new SIP header field (namely Encr-Src) and a new response code (namely 610 Potential Security Breach) to help a UA authorize the security proxy server that encrypts the SIP bodies on behalf of the user [Umschaden03].

Note that Encr-Src header field and 610 code have been already defined in the Internet draft [Umschaden03]. I just quoted their syntax in Appendix 1 for completeness.

6.3.3 Protocol Architecture

Figure 6-3 shows a typical architecture based on this security mechanism. In IPv6 stub domain there is an extension aware UA and SIP security proxy server that is coupled with a NAT-PT box (containing SIP-ALG and DNS-ALG) standing on the boundary between IPv6 stub domain and the public (un-trusted) Internet over IPv4. In the Internet, there exists a domain that includes a UA with a public IPv4 address and a SIP security proxy that supports this extension.

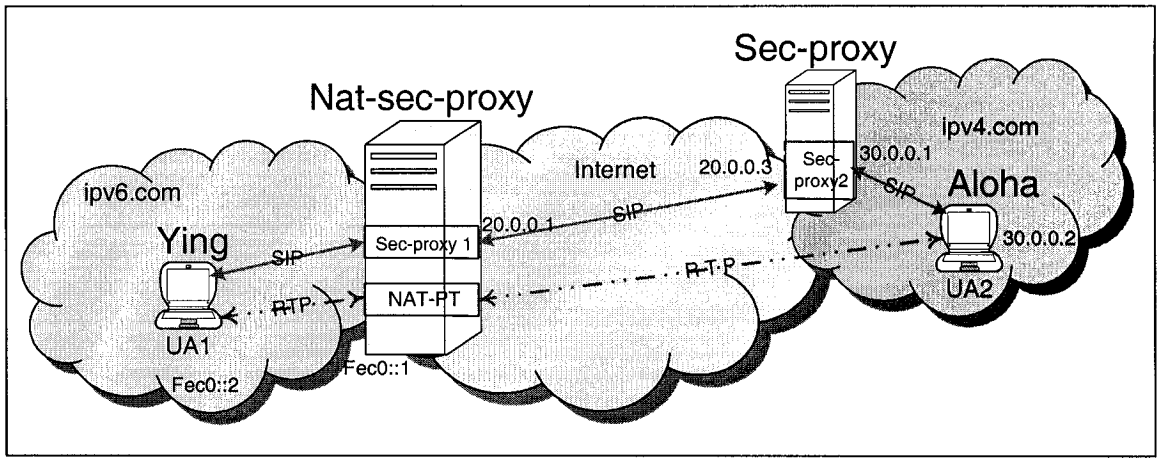

Figure 6-3 End-to-end Security Architecture (NAT-PT/SIP-ALG)

The signaling path from the UA to the SIP proxy must be secured to maintain the desired end-to-end security that occurs between the two security proxies in this architecture. In addition, the security proxy in the IPv4 domain may be co-located with another NAT device that translates global IPv4 addresses to private IPv4 addresses for an intranet.

In this architecture, I assume that the NAT-PT box has enough global IPv4 addresses for translation, which means transport port number does not need to be translated as a packet traverses the IPv4/IPv6 boundary. Hence, RTP media stream can flow via the NAT-PT module without a specific application level gateway.

6.3.4 Protocol Operation

This section illustrates this end-to-end security mechanism with the SIP extension in more detail. There are two steps to establish a SIP dialog on the end-to-end security.

First of all, the initiation and registration of a UA will be described. Secondly, I will explain the session establishment.

6.3.4.1 UA Initiation and Registration

When a UA starts up, it attempts to register at its registrar server. Because all the signaling messages must traverse through the security proxy to enable it to investigate the SDP to provide end-to-end security, the security proxy must be registered as a contact address for the UAs. Also, it is recommended that the security proxy act as a SIP registrar for security reasons.

The detailed description of the secured registration process is introduced in Appendix 2.

6.3.4.2 Session Establishment

Figure 6-4 illustrates the message flow of SIP session establishment according to the architecture (Figure 6-3).

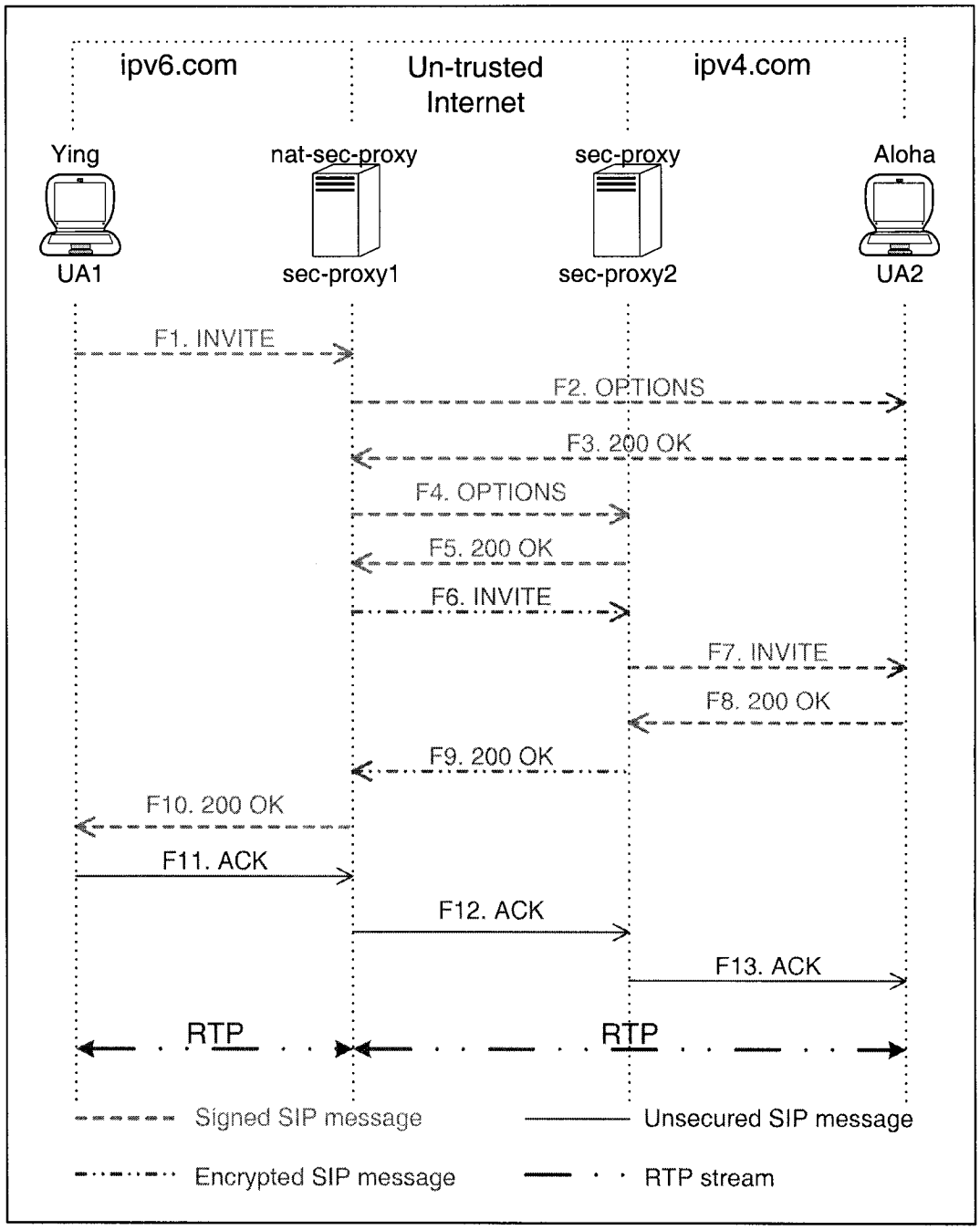

Figure 6-4 SIP Message Flow for Session Establishment

An IPv6 UA authorizes its security proxy by adding a new header field Encr-Src to its INVITE request. This header field contains only the server name of the proxy that is authorized to encrypt the SDP on behalf of the UA. The hostname value in the header

fields must match the common name attribute of the subject in the proxy's S/MIME server certificate that has been received during registration. In addition, the UA is responsible for the tunneling of the integrity of this header field to the receiving UA.

The sending security proxy will discover the receiving UA's capabilities and configuration. Furthermore, it modifies the INVITE message (to protect privacy of the sending domain) and queries SIP-ALG for address translation. Afterwards, it encrypts the SDP for the security proxy in the receiving domain.

As the header fields that are responsible for routing must not be secured, each SIP proxy in the Internet can forward the request downstream until it arrives at the security proxy in the receiving domain. This proxy decrypts the SDP and parses the server name of the security server that encrypted the SDP in order to encrypt its SDP of the response for the sending security server. After this, it forwards the message to the receiving UA by a secure connection.

The receiving UA must check the integrity of the sending security proxy to make sure that the SDP message was not modified by any un-trusted third party. After this, the receiving UA generates a 200 OK response containing an Encr-Src header field that authorizes its security proxy to encrypt the SDP of the response. It also tunnels this header field for the integrity.

The security proxy in the receiving domain encrypts the SDP for the security proxy of the caller domain and forwards the response upstream.

The security proxy in the caller domain decrypts the SDP, retrieves the NAT bindings, translates IP addresses in the response and forwards it to the sending UA.

Upon checking the integrity of the receiving security proxy, the sending UA generates an ACK message and transmits it to the receiving UA in an unencrypted manner.

A detailed example for the secured session establishment is described in Appendix 3.

6.4 Security Considerations

It is necessary to analyze both the contributions and the possible security holes that this end-to-end security mechanism introduces.

Because this mechanism requires that the registration is based on a secure path and Digest authentication to the UAs who originate the request, the Registration Hijacking introduced in Section 5.1.1 can be prevented efficiently. Meanwhile, these requirements could reduce the attack of Denial of Service.

With this security mechanism, SDP is encrypted as transmitting between two security proxies, and the UA is responsible for validating the integrity of the SIP messages. Although some header fields that provide routing capability may be viewed by an attacker in the un-trusted domains, the Impersonating a Server attack or Tampering with Messages attack, such as the man-in-the-middle attack, will be found at the security proxies because the attacker does not have a valid certificate corresponding to the server name in the signed Encr-Src header filed. No one is able to forge SDP because it is encrypted.

Together with the Encr-Src header field the UA must tunnel the Date header field, which helps to prevent an authorized server from re-using the authorization for other requests it is not authorized to use anymore, e.g., some kind of replay attack.

By hiding/restoring the sensitive information of UAs at the security proxy, some UA's privacy and network configuration behind the proxy will be protected.

However, this end-to-end security mechanism also faces some security holes.

The largest outstanding defect with the S/MIME mechanism is the lack of a prevalent public key infrastructure for end users, which is inherited from the defect of S/MIME itself. If self-signed certificates, or certificates that cannot be verified by one of the participants in a dialog are used, the SIP-based key exchange mechanism is susceptible to a man-in-the-middle attack by which an attacker can potentially inspect and modify S/MIME bodies. The attacker needs to intercept the first exchange of keys between the two parties in a dialog, remove the existing CMS-detached signatures from the request and response, and insert a different CMS-detached signature containing a certificate supplied by the attacker. Each party will think they have exchanged keys with the other, when in fact each has the public key of the attacker. However, the attacker has to remain in the path of all future dialogs between the two parties over time. In this end-to-end security mechanism, UAs and security proxies are required to check the validation of the certificates of the senders whenever they receive the SIP messages. If the certificates pose a problem, the end users will decide to continue the call

establishment or accept the call. For this reason, it would be difficult for the attacker to remain online.

This security mechanism depends greatly on a trust relationship to the security proxy, which might be actively malicious. For example, it may eavesdrop on the media stream by modifying SDP and pointing RTP media streams to a wiretapping device. In addition, although the Date header field can help a user to revoke the authorization from the security proxy, it would be possible for the security proxy to use the authorization to set up another simultaneous session with the authorization of the user. Therefore, the security proxy must be very trustful.

Because each intermediary is able to record signaling messages when they pass, it is possible to prematurely tear down a session by forging a BYE request forwarding to the targets of the session.

7 Simulation and Verification of Secure SIP

This chapter will introduce the simulation and verification for both SIP end-to-end security mechanisms mentioned in Section 6.1; one is SIP end-to-end security for Firewall/NAT traversal proposed by Umschaden [Umschaden03]; the other one is for NAT-PT traversal, and is detailed in Section 6.2 through 6.4 in this thesis. Firstly, I will introduce the modeling language Promela and modeling tool Spin. Section 7.2 will model Umschaden's proposal for protocol verification. Finally, I will describe my proposal and detail the approach of modeling the system and protecting against a Man-in-the-Middle (MitM) attack.

7.1 Spin and Promela

Spin [Spin03] is a popular software tool that can be used for the formal verification of distributed software systems. The tool was developed at Bell Labs in the original Unix group of the Computing Sciences Research Center, starting in 1980. The software has been available freely since 1991, and continues to evolve to keep pace with new developments in the field.

Spin can be used in three basic modes [Spin03]:

- as a simulator, allowing for rapid prototyping with random, guided, or interactive simulations.

- as an exhaustive verifier, capable of rigorously proving the validity of user specified correctness requirements.
- as a proof approximation system that can validate even very large protocol systems with maximal coverage of the state space.

All software created by Spin is written in ANSI standard C, so it can be used on different operating systems, such as across all versions of Unix, Linux, Plan9, Inferno, Solaris, Mac, and Windows operating systems.

Spin uses a high level language to specify systems descriptions, called Promela (PROcess MEta LAnguage). "Spin has been used to trace logical design errors in distributed systems design, such as operating systems, data communications protocols, switching systems, concurrent algorithms, railway signaling protocols, etc. The tool checks the logical consistency of a specification. It reports on deadlocks, unspecified receptions, flags incompleteness, race conditions, and unwarranted assumptions about the relative speeds of processes" [Spin03].

Promela provides a means for making abstractions of protocols that suppress details that are unrelated to process interaction. Promela programs consist of processes, message channels, and variables. Processes specify the protocol behavior; channels and global variables define the environment in which the processes run. Processes are global objects. Message channels and variables can be declared either globally or locally within a process.

Given a model system specified in Promela, Spin can perform random or interactive simulations of the system's execution or it can generate a C program that performs a fast

exhaustive verification of the system state space. During simulations and verifications, Spin checks for the absence of deadlocks, unspecified receptions, and un-executable code.

The easiest way to start using Spin is to use the graphical interface Xspin, which runs Spin in the background and generates the proper Spin commands based on menu selections. In addition, Xspin generates a graphical representation for the desired output of Spin, e.g., MSC (Message Sequence Chart).

In brief, completing a protocol model will follow the procedures below:

- Using Promela to specify the protocol and make sure the syntax is error-free.
- Using Spin or Xspin to simulate the system and modify the system if necessary.
- Running Verification tool of Xspin or creating an executable program to verify the modeled system.

7.2 Model SIP End-to-end Security with Firewall/NAT

For comparing the two proposals and complementing the Umschaden's proposal, I have simulated and verified his end-to-end security mechanism using Promela and Spin.

7.2.1 System Architecture and Message Sequence Diagram

Figure 7-1 illustrates the basic architecture and main message sequence flows for Umschaden's proposal. The figure is generated based on Figure 1 and Figure 3 in [Umschaden03]. I have ignored some signaling messages that do not affect the system's main functionality and supplemented or detailed some messages for completeness. For

example, message F6 in [Umschaden03] is detailed into messages F6-1 and F6-2, and the BYE message and FW_CLOSE are supplemental. Please refer to [Umschaden03] for details about the end-to-end security with Firewall/NAT.

Here, I only introduce the changes to [Umschaden03]:

- F6-1. ADDR_REQ

After receiving the INVITE message from the sender, the security proxy Sec-proxy1 will first query the location (i.e., SIP URI) of the destination from a SIP registrar server and the IP address and port number of the URI from DNS services [RFC3263]. After that, it will ask the Firewall/NAT box to reserve NAT bindings and open a firewall pinhole for incoming RTP stream by sending a MIDCOM message named ADDR_REQ. The address and port number carried in the ADDR_REQ to open the pinhole will be obtained from the SDP part of the INVITE message.

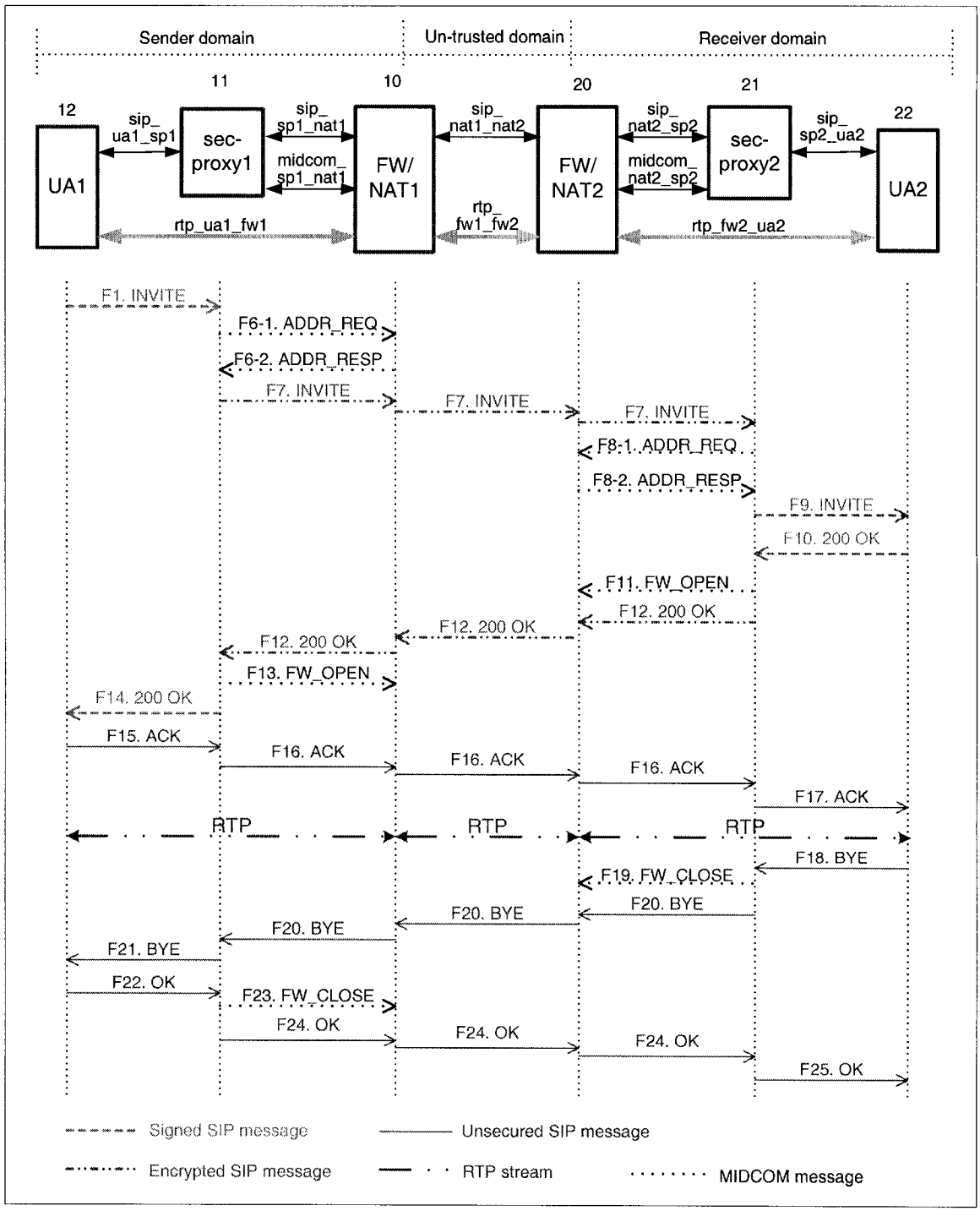

Figure 7-1 Basic Architecture and Message Sequences

- F6-2. ADDR_RESP

Upon receiving the ADDR_REQ, the Firewall/NAT box FW/NAT1 will assign address binding between the private IPv4 addresses and the public ones and return the

bindings to Sec-proxy1 with the ADDR_RESP message. In addition, the private UDP port numbers for incoming RTP stream will be mapped to the global ones. These bindings will be reserved in FW/NAT1.

Note that when the security proxy sends DNS queries, the so-called DNS-ALG using for NAT-PT traversal is not necessary for NAT traversal, but an address mapping in the NAT module will be maintained for the DNS queries.

Sec-proxy1 will substitute the private IP address and port numbers in the SDP with the assigned global ones according to the ADDR_RESP.

- F8-1. ADDR_REQ

Sec-proxy2 receives the SIP messages via the well-known port 5060. After decrypting the SDP, it retrieves the destination information and RTP port number. Moreover, it will query its SIP registrar server for the location of the receiver. After this, it will send an ADDR_REQ for address bindings and opening a firewall pinhole.

- F8-2. ADDR_RESP

FW/NAT2 will create and maintain the mappings for IP addresses and UDP port number. It also opens a pinhole for the outgoing RTP flow. The address bindings will be returned to Sec-proxy2 by ADDR_RESP for address translation.

- F11. FW_OPEN

Sec-proxy2 uses FW_OPEN to ask FW/NAT2 to open a pinhole for incoming RTP messages according to the SDP of the received 200 OK response.

- F13. FW_OPEN

This FW_OPEN is used to ask FW/NAT2 to open a pinhole for outgoing RTP messages according to the SDP of the received 200 OK response.

- F19. and F23. FW_CLOSE

FW_CLOSE is used to close both incoming and outgoing pinholes of RTP stream.

It is worth noting that the MIDCOM messages listed above have not been secured in Umschaden's solution. In addition, there has been no MIDCOM security mechanism introduced so far. To achieve the Umschaden's proposal, the security proxy servers must have already had a trusting relationship with the middleboxes.

7.2.2 Simulation and Verification

Xspin is used as a tool to simulate the system. Simulation is used to analyze the behavior of the model and to check whether the model is equivalent to the requirements in the high level specifications. This section will introduce some details about simulation and verification to the Umschaden's Internet Draft using Promela and Spin.

1. Assumption. To model the main behavior of the end-to-end security system illustrated in Figure 7-1, some assumptions should be made to ignore the irrelevant behaviors. First, the registration procedures are ignored in the modeling system because the system is focusing on the end-to-end security signaling negotiation. Second, I assume the security algorithm of S/MIME recommended by SIP Request For Comments [RFC3261] would work well in the security proxy, which means the encryption algorithm

will not affect the main behavior of the end-to-end security mechanism. Hence, the encryption algorithm is ignored in this model. However, a variable is necessarily used to indicate whether the SDP has been encrypted or not. SIP messages ACK and BYE/200 are not secured because they do not carry message bodies that should be encrypted, and because the security proxies are stateful, while there is no response message for ACK.

2. Specification. Promela specifies a system by using processes, message channels, and variables. In this system, there are six components, each of which is specified by a process. That is, the component 'UA1' is specified by the process 'UA1', 'sec-proxy1' by 'sec_proxy_1', 'FW/NAT1' by 'FW_NAT_1', 'FW/NAT2' by 'FW_NAT_2', 'sec-proxy2' by 'sec_proxy_2', and 'UA2' by 'UA2'. The system defines three kinds of message channels: 'sip_', 'rtp_', and 'midcom_' used to transmit SIP, RTP, and MIDCOM messages respectively. Table 7-1 shows the relationship between the protocol components and modeling processes and their channels included. In addition, the system defines three kinds of messages: SIP messages ('INVITE', 'ACK', 'BYE', 'OK'), RTP messages ('RTP_IN', 'RTP_OUT'), and MIDCOM messages ('ADDR_REQ', 'ADDR_RESP', 'FW_OPEN', 'FW_CLOSE').

COMPONENT	PROCESS	CHANNELS
UA1	UA1	sip_ua1_sp1, rtp_ua1_fw1
sec-proxy1	sec_proxy_1	sip_ua1_sp1, sip_sp1_nat1, midcom_sp1_nat1
FW/NAT1	FW_NAT_1	sip_sp1_nat1, sip_nat1_nat2, rtp_ua1_fw1, rtp_fw1_fw2, midcom_sp1_nat1
FW/NAT2	FW_NAT_2	sip_nat1_nat2, sip_nat2_sp2, rtp_fw1_fw2, rtp_fw2_ua1, midcom_nat2_sp2
sec-proxy2	sec_proxy_2	sip_nat2_sp2, sip_sp2_ua2, midcom_nat2_sp2
UA2	UA2	sip_sp2_ua2, rtp_fw2_ua2

Table 7-1 System Specification for Umschaden's Solution

3. Simulation. Given the modeling system specified by Promela, Spin is used to perform a random or interactive simulation of the system's execution. Figure 7-2 is a MSC output created by Xspin corresponding to the architecture shown in Figure 7-1. In Figure 7-2, the information with dashed arrows are comments for understanding the figure. Comparing the two figures, we can see that the behavior of the model is equivalent to the requirements in the protocol specifications.

4. Verification. After simulation, Spin can run an exhaustive verification. The model has passed the Exhaustive Verification by Spin. From the output of the verification, the system has no deadlocks and unspecified receptions. There are some

un-executable codes, but they are error-handlers in the program written in Promela. Therefore, the result of the verification proves that the Umschaden's solution is feasible.

Note that the model will not run to completion with the default value (1024) of the parameter VECTORSZ. To achieve the exhaustive verification, the value must be reset, for example, to 2048. There are two ways to do this:

- Modify the file pan.h.

  Locate to the line:

  ```
  #define VECTORSZ   1024          /* sv     size in bytes */
  ```

  Change 1024 to 2048 or more.

- In the command line, run:

  ```
  gcc –o <output filename> -DVECTORSZ=2048 pan.c
  ```

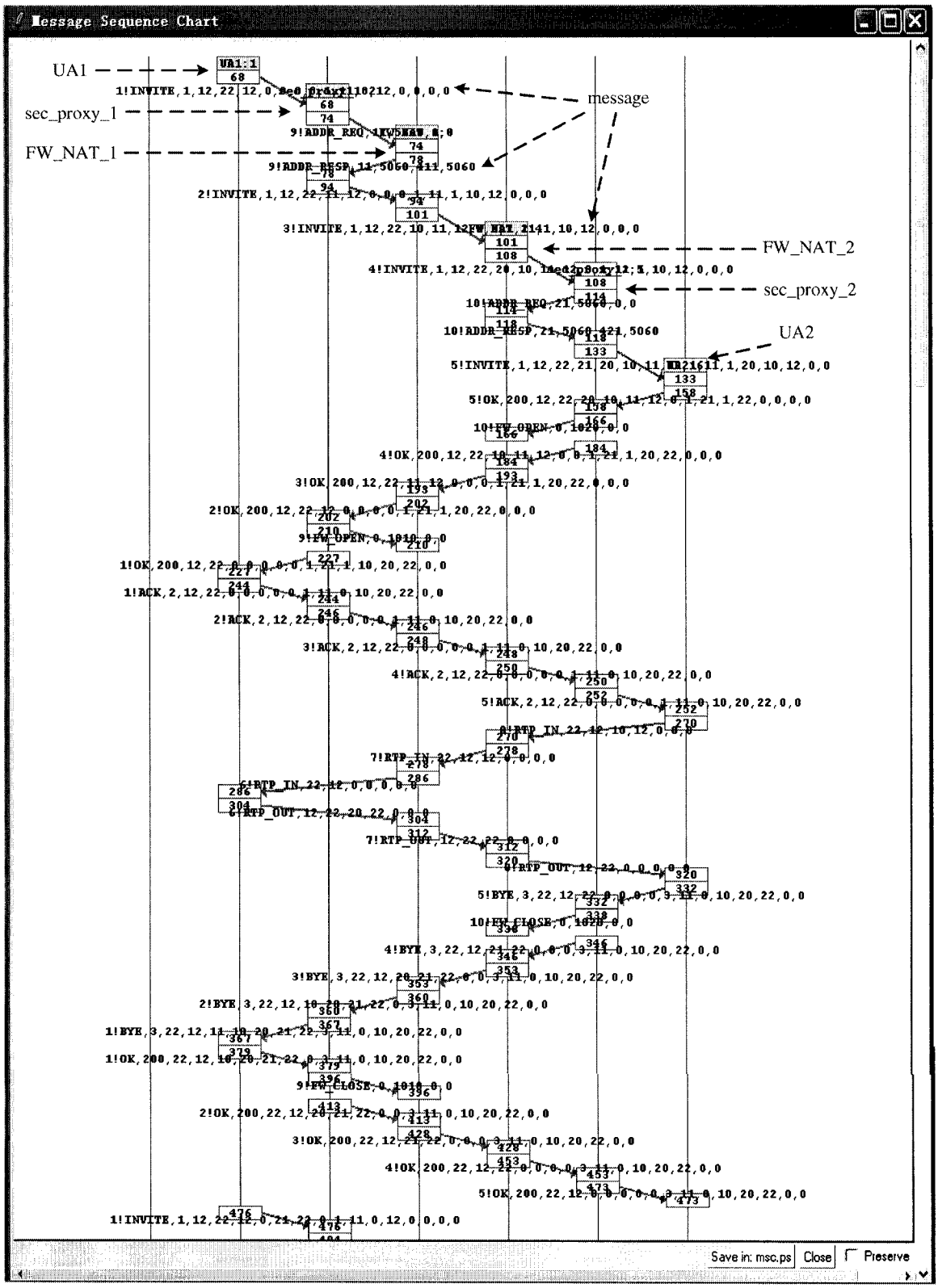

Figure 7-2 MSC for the end-to-end Security with Firewall/NAT

7.3 Model SIP End-to-end Security with NAT-PT

This section will describe simulation and verification of the SIP end-to-end security with NAT-PT traversal. A Man-in-the-middle attack is added into the system for comparing enabling/disabling the security mechanism.

7.3.1 System Architecture and Design

The detailed SIP protocol specification and SIP end-to-end security mechanism has been described in the previous sections. The system to be modeled is based on Figure 6-3, and the system adds another normal SIP proxy server in the un-trusted Internet. This proxy will have two functions in the modeled system:

- This proxy is responsible for routing SIP messages to their targets as a normal SIP proxy server does.
- This proxy acts as a Man-in-the-Middle (MitM) attacker to redirect the RTP media stream to itself for eavesdropping the dialog by modifying the SDP message within the SIP messages traversing through it. In detail, the MitM attacker will check the SIP header fields to route the SIP messages to the next hop. Also, he could check the SDP part for malicious purposes. If the SDP is encrypted, he can do nothing because he is not capable of decrypting the message. However, if the end-to-end security mechanism does not exist, then the SDP message is plain text. In this case, the attacker can modify the SDP message in order to make the future RTP stream go through its wiretapping device.

Figure 7-3 illustrates the system architecture when the SIP end-to-end security mechanism enables, while Figure 7-4 shows the system architecture in which there is no security mechanism, i.e., the attacker achieves his purpose.

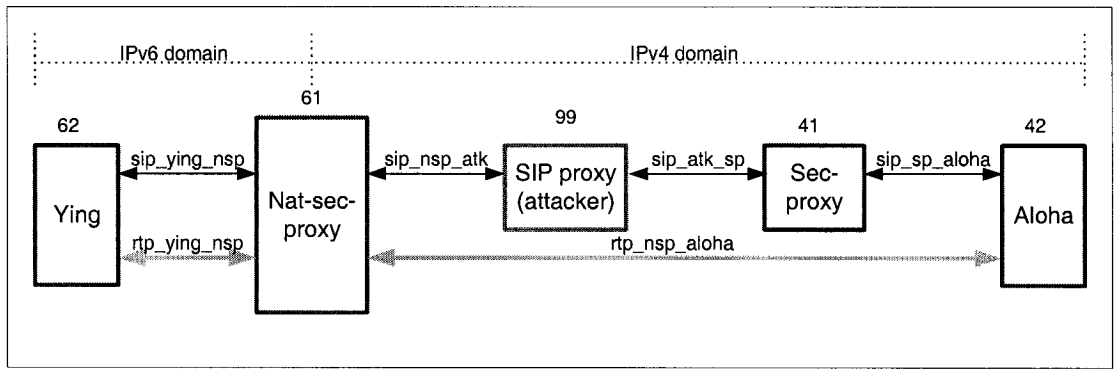

Figure 7-3 System Architecture with Security Mechanism

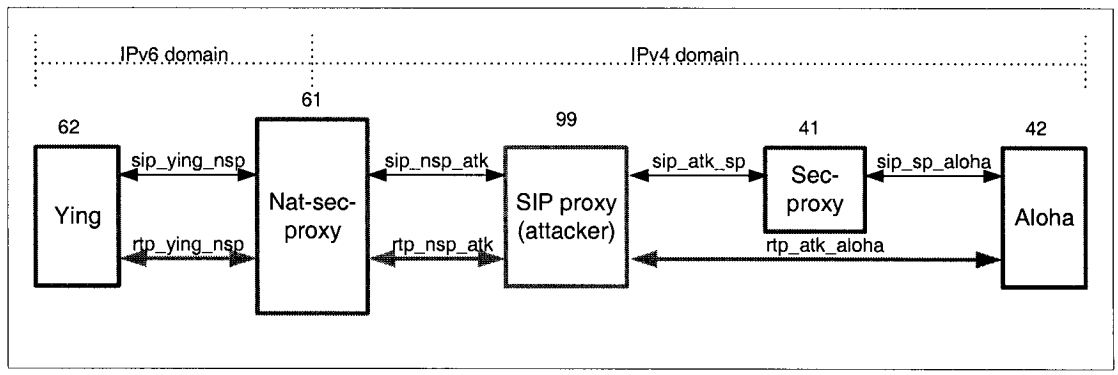

Figure 7-4 System Architecture without Security Mechanism

7.3.2 Simulation and Verification

This section will introduce details about simulation and verification for the solution proposed in this thesis using Promela and Spin.

1. Assumption. I have modeled the main behavior of the SIP end-to-end security system with some assumptions. First, the IPv6/IPv4 address translation has been implemented, and it transparently provides services to the security proxy; therefore, I do

not need to model it in the system. Second, the specification of S/MIME is ignored for the same reason as that of the modeling system of the Umschaden's Internet Draft. Similarly, the registration procedures are ignored in the modeling system, and the SIP messages ACK and BYE/200 do not need to be secured.

There is a scenario in the modeled system both with and without the end-to-end security mechanism:

1) Ying initiates a SIP dialog using an INVITE request;
2) Aloha responds a 200 OK response;
3) Ying confirms the response by an ACK request;
4) Aloha and Ying establishes a RTP media stream;
5) Aloha tears down the dialog using a BYE request;
6) Ying acknowledges a 200 OK.

2. Specification. In the system, there are five entities: Ying, Nat-sec-proxy, SIP proxy (attacker), Sec-proxy and Aloha, each of them specified as a process by Promela: 'Ying', 'nat-sec-proxy', 'attacker', 'sec-proxy' and 'Aloha' respectively. The system specifies three kinds of message channels: 'sip_', 'rtp_', and 'midcom_' used to transmit SIP, RTP, and MIDCOM messages respectively. Table 7-2 shows the relationship between the protocol components and modeling processes and their channels included. All message channels are defined as rendezvous (synchronous) channels. In addition, the system defines two kinds of messages: SIP messages ('INVITE', 'ACK', 'BYE', 'OK'), and RTP messages ('RTP_IN', 'RTP_OUT').

COMPONENT	PROCESS	CHANNELS
Ying	Ying	sip_ying_nsp, rtp_ying_nsp
Nat-sec-proxy	nat_sec_proxy	sip_ying_nsp, sip_nsp_atk, rtp_ying_nsp, rtp_nsp_aloha, rtp_nsp_atk
SIP proxy (attacker)	attacker	sip_nsp_atk, sip_atk_sp, rtp_nsp_atk, rtp_atk_aloha
Sec-proxy	sec_proxy	sip_atk_sp, sip_sp_aloha
Aloha	Aloha	sip_sp_aloha, rtp_nsp_aloha, rtp_atk_aloha

Table 7-2 System Specification for My Proposal

3. Simulation. Given the modeling system specified by Promela, Spin is used to perform a random or interactive simulation of the system's execution. Figure 7-5 is a MSC output created by Xspin corresponding to the architecture shown in Figure 7-3. Figure 7-6 is the output for the architecture without the security mechanism (Figure 7-4). In Figure 7-5 and Figure 7-6, the information with dashed arrows or circle is comments. Notice that the difference between the two figures is as such: in Figure 7-5, RTP stream goes through Ying<->nat-sec-proxy<->Aloha; in Figure 7-6, the stream goes via Ying<->nat-sec-proxy<->attacker<->Aloha. The reason for this difference is that in Figure 7-5 the SDP message is encrypted, so the attacker cannot modify the SDP message; while in Figure 7-6 (without security mechanism) the attacker is able to act as an unexpected gateway to forward the RTP media stream by modifying the SDP message.

4. Verification. After simulation, Spin performs an exhaustive verification. The model has passed the Exhaustive Verification by Spin. From the output of the verification, the system has no deadlocks and unspecified receptions. There are some un-executable codes; however, some of them are error-handlers in the program written in Promela, and the others correspond to the different selection of enabling/disabling the end-to-end security mechanism. Therefore, the SIP end-to-end security architecture should work well.

To achieve the exhaustive verification, the value of the parameter VECTORSZ must be reset, for example, to 2048.

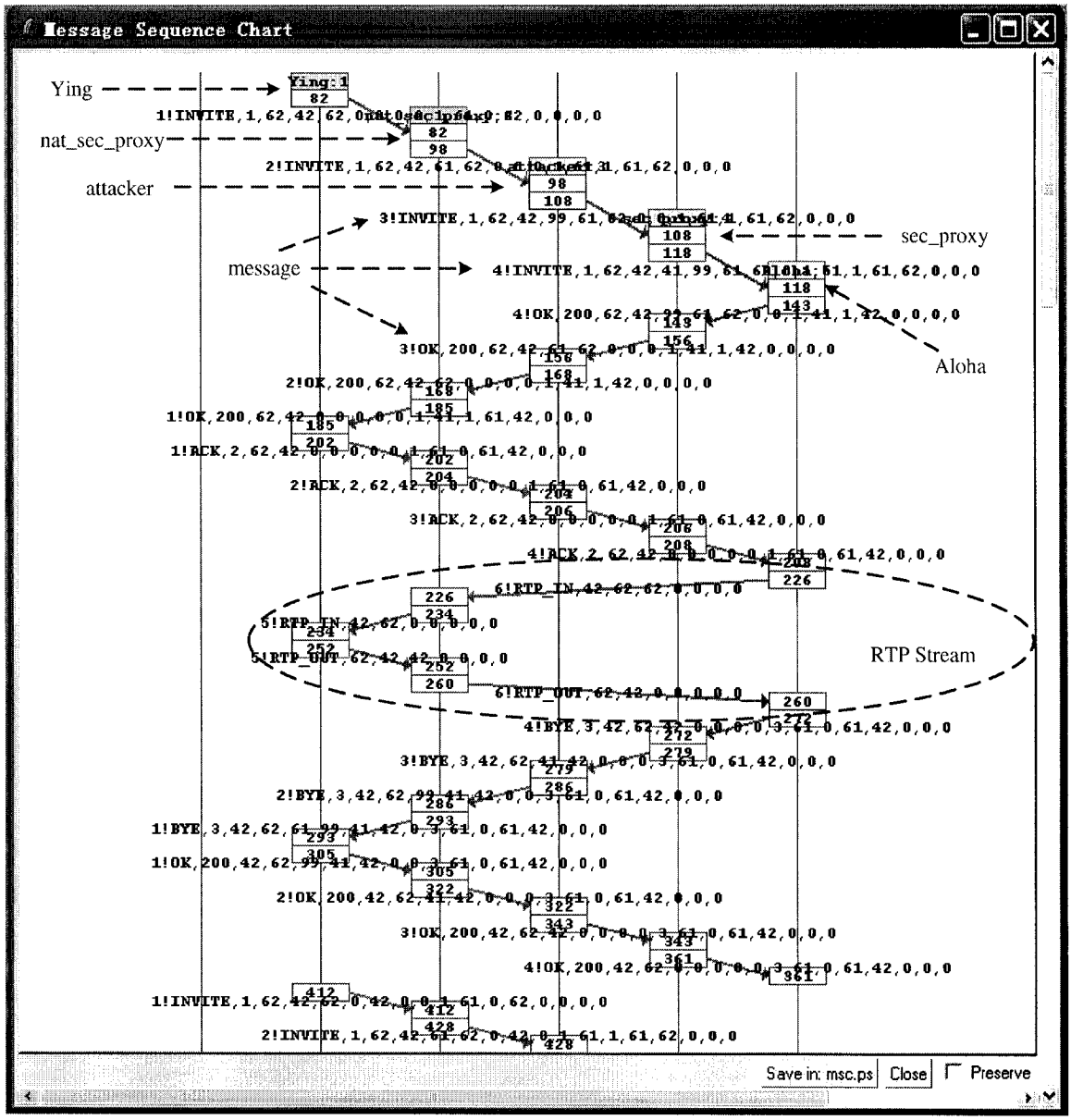

Figure 7-5 MSC for the System with Security Mechanism

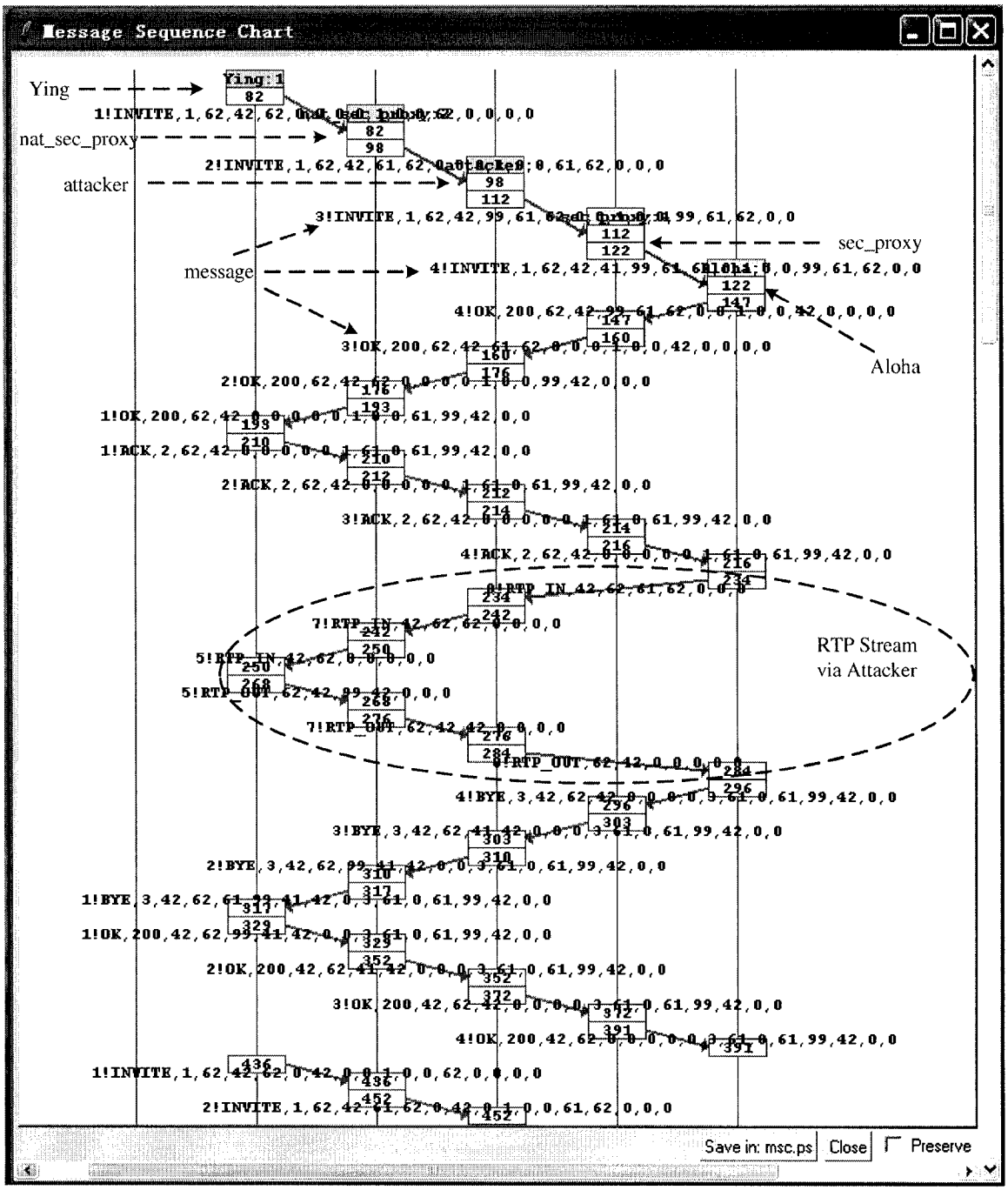

Figure 7-6 MSC for the System without Security Mechanism

8 Conclusion and Discussion

This thesis has solved two problems surrounding IP address transition and the SIP protocol by providing the solutions of implementing a SIP-ALG and building an architecture for SIP end-to-end security mechanism.

SIP-ALG is a specific application level gateway for SIP sessions that is responsible for translating IPv6 addresses in a SIP packet into the corresponding IPv4 ones, and vice versa, relying on the functions of NAT-PT as the packet traverses the boundary between IPv6 and IPv4. NAT-PT is capable of maintaining address mapping between IPv6 addresses and IPv4 addresses and transiting IP header part, but not dealing with the IP payloads. Once IP payloads are built by the SIP protocol, a SIP-ALG is necessary, since the IP payloads, i.e., SIP messages, may contain IP address in both SIP header fields and bodies, e.g., SDP messages. SIP-ALG translates the IP addresses according to the address mapping already built by NAT-PT. In addition, a DNS-ALG is needed for most applications such as SIP sessions due to name-address query. However, it might be unnecessary if no name query is needed. For example, if a SIP call is established by raw IP addresses, the DNS-ALG keeps silent in my experiments.

Based on the IP address transition by SIP-ALG, the SIP end-to-end security mechanism enables end-to-end security of SIP bodies, i.e., SDP. This solution depends on S/MIME and TLS. An end user authorizes a security proxy server by an extended SIP header field in order to encrypt the SDP messages on behalf of the user. The security

proxy will discover the capabilities of the receiving party and encrypt the SDP messages for the other SIP security proxy server in the receiving domain. IP address translation must be done before the encryption. As long as each end user connects its trustful security proxy via a TLS connection and authorizes this proxy to encrypt the SDP messages by S/MIME, the session information is secured end-to-end. Here, TLS is not mandatory. The connection between the end user and the security proxy may be built by any security means, such as IPSec.

Moreover, I would like to discuss some possible applications and extensions based on this end-to-end security mechanism.

Firstly, referring to the architecture illustrated in Figure 6-3 in Section 6.3.3, the IPv4 end point has a public IPv4 address. More often end users, however, are located in a local LAN that makes use of private IPv4 address for each user and employs NAT devices in the boundary to the Internet. In this case, a SIP-ALG is required to translate between the private IPv4 addresses and the mapped public IPv4 addresses. In addition, the SIP security proxy for this LAN should be co-located with the NAT device and the SIP-ALG.

Secondly, to make the proposed end-to-end security mechanism more complicated, the Network Address Port Translation – Protocol Translation (NAPT-PT) could be used to replace the NAT-PT.

The main disadvantage of NAT-PT is that it has to own enough public IPv4 addresses in its address pool for mapping a potentially large number of IPv6 end points. NAPT-PT could solve this problem. It allows IPv6 nodes to communicate with the IPv4 nodes

transparently using a single IPv4 address, but with different TCP/UDP ports. In principle, one IPv4 address can support 2^{16} address-port mappings. The TCP/UDP ports of the IPv6 nodes are translated into TCP/UDP ports of the assigned IPv4 address at the NAPT-PT module.

However, the NAT-PT and ALGs must be updated properly. The NAPT-PT module must be able to maintain address-port mapping, instead of only address mapping in NAT-PT module. In addition, it must translate transport identifiers, i.e., TCP/UDP ports in IP payloads. As for ALGs, they also must analyze and modify transport identifiers in the corresponding messages, such as SIP messages. Note that RTP protocol still does not need ALG because no IP address or UDP port number is contained in RTP messages.

Thirdly, the NA(P)T-PT box including ALGs and the SIP security proxy will usually be located behind a firewall for enterprise security reasons. In this case, the NA(P)T-PT module would be able to control the firewall and open/close the pinholes of the firewall according to the address mapping. To control the firewall, the firewall could be a logical module and be co-located with the NA(P)T-PT box. Thus, the NA(P)T-PT including ALGs and the security proxy will be a large package. In order to facilitate network administration, the package should be divided into several parts, each part for one functionality, such as NA(P)T-PT, DNS-ALG, SIP-ALG, firewall, and SIP security proxy. Moreover, each part should establish communications for controlling purpose by some signaling protocols, for example the Middlebox Communication (MIDCOM) protocol [RFC3303, RFC3304], for which deployment is in progress. Using MIDCOM, the

address translation can support the end-to-end security mechanism transparently. That is, the upgrades from NAT module to NA(P)T-PT module will cause few changes of other entities, such as security proxy servers and Firewalls.

Lastly, for 3G mobile users, the tunneled and encrypted SIP packets might be too large; the proper compression/decompression algorithms for MIME should be considered.

Bibliography

[Atwood03-1] J. William Atwood, Xing Jiang, Kedar C. Das. *Establishing Multimedia Sessions between IP version 6 Hosts and IP version 4 Hosts*. Proceedings of CSA 2003, pp. 538-542. Banff, Alberta, Canada, July 2003.

[Atwood03-2] J. William Atwood, Kedar C. Das, Xing Jiang. *IPv4/IPv6 Translation*. Proceedings of the 2003 Linux Symposium, pp. 39-48. Ottawa, Ontario, Canada, July 2003.

[Bajko01] G. Bajko, B. Bertenyi and K. Kiss. *Multimedia Sessions Between 3G Wireless and Internet Users*. Nokia Networks Hungary. January 2001.

[Flykt01] P. Flykt and T. Alakoski. *SIP Services and Interworking with IPv6*. Second International Conference on 3G Mobile Communication Tech-nologies, pp.186-190, March 2001.

[IPv603] IPv6style.jp. *Learning about the IPv6 header*. http://www.ipv6style.jp/en/tech/20030331/index.shtml. March 2003.

[Kristensen02] L. K. Kristensen and K. Nielsen. *NA(P)T-PT Translation within the WCDMA/GSM Network*. Ericsson Research. October 2002. (In progress)

[Madrid00] Juan M. Madrid. *Multimedia Information Transport over the Internet: The RTP Protocol*. University of Kansas, April 2000.

[Natpt01] Next Generation Internet Standard Research Team. *Linux-based Userspace NAT-PT*. http://www.ipv6.or.kr/english/download.htm . Electronics and Telecommunications Research Institue. May 2001.

[Radvision01] *SIP: Protocol Overview*. Radvision Ltd. 2001.

[Rahamim02] U. Rahamim. *Perspective: IPv6, the Net's next frontier*. http://news.com.com/2010-1069-976872.html, December 2002.

[RFC1847] J. Galvin, S. Murphy, S. Crocker and N. Freed, *Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted*, Request For Comments 1847, Internet Engineering Task Force, October 1995.

[RFC1889] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications*. Request For Comments 1889, Internet Engineering Task Force, January 1996.

[RFC2234] D. Crocker and P. Overell (Ed.). *Augmented BNF for Syntax Specifications: ABNF*. Request For Comments 2234, Internet Engineering Task Force, November 1997.

[RFC2246] T. Dierks and C. Allen. *The TLS Protocol Version 1.0*. Request For Comments 2246, Internet Engineering Task Force, January 1999.

[RFC2326] H. Schulzrinne, A. Rao and R. Lanphier. *Real Time Streaming Protocol (RTSP)*. Request For Comments 2326, Internet Engineering Task Force, April 1998.

[RFC2327] M. Handley and V. Jacobson. *SDP: Session Description Protocol*. Request For Comments 2327, Internet Engineering Task Force, April 1998.

[RFC2401] S. Kent and R. Atkinson. *Security Architecture for the Internet Protocol*. Request For Comments 2401, Internet Engineering Task Force, November 1998.

[RFC2402] S. Kent and R. Atkinson. *IP Authentication Header*. Request For Comments 2402, Internet Engineering Task Force, November 1998.

[RFC2406] S. Kent and R. Atkinson. *IP Encapsulating Security Payload (ESP)*. Request For Comments 2406, Internet Engineering Task Force, November 1998.

[RFC2409] D. Harkins and D. Carrel. *The Internet Key Exchange (IKE)*. Request For Comments 2409, Internet Engineering Task Force, November 1998.

[RFC2617] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen and L. Stewart. *HTTP authentication: Basic and Digest Access Authentication*. Request For Comments 2617, Internet Engineering Task Force, June 1999.

[RFC2630] R. Housley, *Cryptographic Message Syntax*, Request For Comments 2630, Internet Engineering Task Force, June 1999.

[RFC2633] B. Ramsdell, *S/MIME Version 3 Message Specification*, Request For Comments 2633, Internet Engineering Task Force, June 1999.

[RFC2766] G. Tsirtsis and P. Srisuresh. *Network Address Translation – Protocol Translation (NAT-PT)*. Request For Comments 2766, Internet Engineering Task Force, February 2000.

[RFC3015] F. Cuervo, N. Greene, A. Rayhan, C. Huitema, B. Rosen and J. Segers, *Megaco Protocol Version 1.0*. Request For Comments 3015, Internet Engineering Task Force, November 2000.

[RFC3261] J. Rosenberg, H. Schulzrinne, G. Camarillo,A. R. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler. *SIP: Session Initiation Protocol*. Request For Comments 3261, Internet Engineering Task Force, June 2002.

[RFC3263] J. Rosenberg and H. Schulzrinne. *Session Initiation Protocol (SIP): Locating SIP Servers*. Request For Comments 3263, Internet Engineering Task Force, June 2002.

[RFC3303] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor and A. Rayhan. *Middlebox communication architecture and framework*. Request For Comments 3303, Internet Engineering Task Force, August 2002.

[RFC3304] R. P. Swale, P. A. Mart, P. Sijben, S. Brim and M. Shore. *Middlebox Communications (midcom) Protocol Requirements*. Request For Comments 3304, Internet Engineering Task Force, August 2002.

[Spin03] Spin home page. http://spinroot.com/spin/whatispin.html. Spinroot.com, 2003.

[Thernelius00] F. Thernelius. *SIP, NAT and Firewalls*. Ericsson, May 2000.

[Umschaden03] K. Umschaden, J. Stadler and I.Miladinovic. *End-to-end Security for Firewall/NAT Traversal within the Session Initiation Protocol (SIP)*. Internet Draft, Internet Engineering Task Force, May 2003.

[Vocal03] VOCAL home page. http://www.vovida.org/, VOVIDA.org. 2003.

Appendix 1 Syntax of SIP Extension

This appendix describes the extended SIP header field (Encr-Src) and the Response code 610 introduced in Section 6.3.2.

1. Encr-Src Header Field

The Encr-Src header field is used by UAs to authorize a proxy server to encrypt and decrypt SDP on behalf of the user. The following syntax specification uses the augmented Backus-Naur Form (BNF) as described in [RFC2234].

```
Encr-Src      = "Encr-Src" HCOLON host [";" encr-flag]
                    *( ";" cert-warning )
encr-flag     = "offensive" / "secure"
cert-warning  = cert-type EQUAL cert-problem
cert-type     = "usercert" / "servercert"
cert-problem  = "expired" / "revoked" / "selfsigned" / "unknowroot"
                    / "mismatch"
```

In this extension for SIP, the values of encr-flag are restricted to the above option. If the encr-flag is present in the Encr-Src header field, it can only take one of the values below:

- offensive: The user requests that the security proxy must try to establish the session, even if the certificates of the callee or the security proxy in the receiving domain are invalid or cannot be validated. If these certificates pose a problem, an appropriate cert-warning will be added to the header field of the result to inform the user about the state of each certificate.

- secure: The user requests that the security proxy must not continue with the session establishment if the security proxy cannot validate a certificate or the certificate is invalid. A 610 Potential Security Breach response will be generated containing the appropriate cert-warning.

A missing encr-flag indicates that the user wants the "secure" solution.

A cert-warning indicates that a problem with a certificate occurred. If a security proxy detects an abnormality with a certificate, it transfers this information to its UA. It is possible that either the certificate of the communication partner or the certificate of the opposite security proxy poses the problem. The cert-type defines which certificate is not OK.

- usercert: The certificate of the communication partner is not OK.
- servercert: The certificate of the security proxy of the communication partner poses problems.

The cert-problem can be one of the following:

- expired: The certificate is expired, it is not valid anymore.
- revoked: The certificate is revoked, therefore it is not valid.
- selfsigned: The certificate is valid, but it is self signed, which indicates that the identity of the certificate holder cannot be determined.
- unknownroot: The certificate is valid, but the root certification authority is not known. Therefore it cannot be determined if the certificate is trustworthy or not. The identity of the certificate holder is not assured.

- mismatch: The server name does not match the common name (CN) attribute of the server certificate subject, or the end-user certificate does not identify the address-of-record of the communication partner, respectively.

Each combination of the cert-type and cert-problem is possible, even multiple cert-warnings can occur. A UA receiving a request or response that contains a security warning must inform the user about the security risk of the broken certificate chain.

Note that the integrity check of this header field must be performed in the following manner: all header field parameters, i.e., encr-flag and cert-warning must be removed. The result will be just the header field name and the hostname of the authorized server. After removing the header field name, which is case-insensitive, the result is solely the hostname. This value must be compared simply byte-by-byte.

This extension does not support encryption of SDP in provisional responses, ACK and UPDATE requests simply because they do not carry SDP.

2. Response Code 610 (Potential Security Breach)

This response indicates that there has been a potential security hazard that cannot be detected by the UA itself. Therefore, this message is returned to the caller. The response 610 carries at least one cert-warning in the Encr-Src header field.

This status response does not indicate in any sense whether a communication partner is willing to accept a request or not, it just outlines potential security problems.

Appendix 2 Secured Registration Procedures

This appendix describes an example of the secured SIP registration process corresponding to Section 6.3.4.1.

Figure A2-1 shows the message flow for a secured registration between a UA and a security proxy.

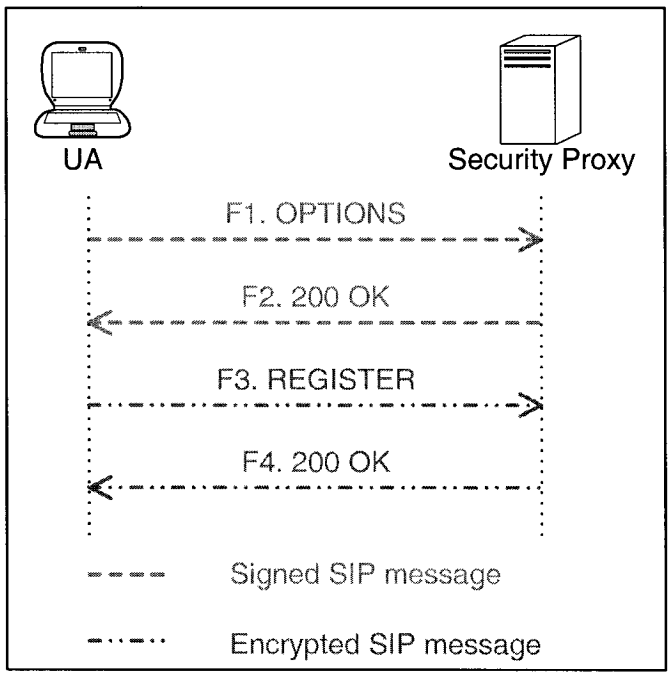

Figure A2-1 Secured SIP Registration

F1. – The UA sends an OPTIONS to discover the capability of the proxy, especially to know whether the proxy supports the Encr-Src field and to obtain the certificate of the proxy if necessary. The UA should structure the body as an S/MIME "multipart/signed" CMS SignedData body, as declared in Section 5.2.4.2.

F2. – The security proxy responds with a 200 OK that contains the server certificate of the proxy in an S/MIME "multipart/signed" CMS SignedData body. The common name (CN) attribute in the subject of the received certificate contains the server name of the SIP security proxy.

If the security server supports the Encr-Src field, it has to indicate this in the Supported header field in this 200 response. This looks like

Supported: Encr-Src

If there is no this kind of Supported field listed above in the response, the UA should assume that the proxy does not support the extension to act as a security proxy and alert the user about this.

Note that if the UA has already holds the certificate of the security proxy and knows that the proxy supports the extension, the messages F1 and F2 may be omitted.

F3. – The UA sends a REGISTER request to the security proxy in order to register it with the address-of-record from the security proxy. This will ensure that all incoming messages for the UA will traverse the security proxy. The UA should use a secure connection for this registration, such as by TLS.

The security proxy is recommended to challenge the user for authentication before updating the location data.

F4. – After successful registration, the security proxy answers a 200 OK response.

Note that all outgoing SIP messages must be routed via the security proxy for SDP encryption.

For more information about SIP registration, please refer to Section 10 in [RFC3261].

Appendix 3 Example for Secured Session Setup

This appendix describes an example of the session establishment for the end-to-end SIP security mechanism with NAT-PT and SIP-ALG. The messages in this example are corresponding to Figure 6-4 in Section 6.3.4.2.

To begin with, I would like to assume the network configuration. UA1 (Ying) is an IPv6-only user agent located in "ipv6.com" domain. The security proxy Sec-proxy1 is co-located with a NAT-PT box that is a dual stack (fec0::1 for ipv6.com and 20.0.0.1 for IPv4 interface). The IPv4 address pool of the NAT-PT has space 20.0.0.3 ~ 20.0.0.20; the IPv6 address pool uses a prefix abcd::/96. For example, 20.0.0.3 and abcd::30.0.0.2 are both valid. UA2 (Aloha) is an IPv4-only user agent, and its security proxy is Sec-proxy2. They are both located in ipv4.com domain.

Both security proxy servers also are responsible for registration functionality as described in Section 6.3.4.1.

I also assume these four SIP entities supports the extension explained in Section 6.3.2. Moreover, both UAs have already registered on their corresponding security proxy.

Assume that Ying in IPv6 stub domain wants to setup a secure call to Aloha in the Internet.

F1. INVITE (Ying -> nat-sec-proxy)

UA1 will generate an INVITE request in a secure way. It inserts the Encr-Src header filed in the request to authorize its security proxy nat-sec-proxy. The value of the header field should be nat-sec-proxy.ipv6.com. In addition, the UA must tunnel the integrity of Encr-Src header field to the receiving UA (Aloha) in order that Aloha can determine whether the encrypting proxy server was authorized to perform the encryption. Together with the Encr-Src header field the UA must tunnel the Date header field to prevent that nat-sec-proxy can reuse the authorization for other requests it is not authorized anymore. The UA may additionally tunnel other SIP header fields that need end-to-end encryption to the callee.

UA1 must create a "multipart/mixed" content, which carries the SDP ("application/sdp") and a signed "message/sip" MIME body.

Note that SDP is left as a format of plain/text in order that the security proxy can modify it. The SDP will be encrypted at the proxy.

The following example is an incomplete SIP message in which I omitted some unimportant header fields for the purpose of brevity.

F1 INVITE Ying -> nat-sec-proxy

INVITE sip:aloha@ipv4.com SIP/2.0
Via: SIP/2.0/TLS [fec0::2];branch=z9hG4bKnas12f8
To: <sip:aloha@ipv4.com>
From: <sip:ying@ipv6.com>; tag=456222
Call-ID: df49df90s8d
Contact: <sip:Ying@[fec0::2]>
CSeq: 1 INVITE
Date: Thu, 14 Aug 2003 13:23:13 GMT
Encr-Src: nat-sec-proxy.ipv6.com

Route: <sip:nat-sec-proxy.ipv6.com; lr>
Content-Type: multipart/mixed; boundary=outer-boundary

--outer-boundary
Content-Type: application/sdp

v=0
o=Ying 2890844111 2890844222 IN IP6 [fec0::2]
s=Session SDP
c=IN IP6 [fec0::2]
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

--outer-boundary
Content-Type: multipart/signed;
protocol="application/pkcs7-signature";
micalg=sha1; boundary=header-boundary

--header-boundary
Content-Type: message/sip

Date: Thu, 14 Aug 2003 13:23:13 GMT
Encr-Src: nat-sec-proxy.ipv6.com

--header-boundary
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--header-boundary-

--outer-boundary-

F2. OPTIONS (nat-sec-proxy -> Aloha)

The nat-sec-proxy receives the INVITE with the Encr-Src header field containing its hostname. Therefore, it is authorized to modify and encrypt the SDP on behalf of the user Ying.

First, the proxy should discover whether the target UA supports this security mechanism by sending an OPTIONS query. This OPTIONS contains a Require header field indicating that the opposite UA must support the Encr-Src extension. The request should be signed and may be encrypted if the public key of the target user is known. Furthermore, nat-sec-proxy must append its S/MIME server certificate that is not visible because the certificate is included in the CMS SignedData body.

F2. OPTIONS nat-sec-proxy -> Aloha

```
OPTIONS sip:aloha@ipv4.com SIP/2.0
Via: SIP/2.0/TCP nat-sec-proxy.ipv6.com;branch=z9hG4bKnappe1
To: <sip:aloha@ipv4.com>
From: <sip:nat-sec-proxy.ipv6.com>; tag=990099
Call-ID: rs8943jdf804s
Contact: <sip:nat-sec-proxy@ipv6.com>
CSeq: 1 OPTIONS
Date: Thu, 14 Aug 2003 13:23:13 GMT
Require: Encr-Src
Content-Type: multipart/signed;
        protocol="application/pkcs7-signature";
        micalg=sha1; boundary=outer-boundary

--outer-boundary
Content-Type: message/sip

OPTIONS sip:aloha@ipv4.com SIP/2.0
To: <sip:aloha@ipv4.com>
From: <sip:nat-sec-proxy.ipv6.com>; tag=990099
```

Call-ID: rs8943jdf804s
Contact: <sip:nat-sec-proxy@ipv6.com>
CSeq: 1 OPTIONS
Date: Thu, 14 Aug 2003 13:23:13 GMT
Require: Encr-Src

--outer-boundary
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--outer-boundary-

After receiving the OPTIONS request, sec-proxy relays the request to UA2.

F3. 200 OK (Aloha -> nat-sec-proxy)

If Aloha does not support the required Encr-Src extension, it will respond a 420 Bad Extension to nat-sec-proxy and set the value of the Unsupported filed as Encr-Src.

If Aloha supports the extension, it must check whether the server certificate included in the request has been revoked, expired, or the root certification authority is unknown, or mismatch between the common name attribute of the certificate subject and the name of the security proxy. If the certificate check is successful, the UA should return a signed 200 OK response. This response should include a Supported header field containing the Encr-Src tag. Because UA2 is located behind a security proxy, it must append an Encr-Src header field containing the hostname of the server sec-proxy that is authorized

to encrypt the SDP for UA2. The header field value must be identical with the common name attribute of the subject of the sec-proxy's server certificate.

UA2 should tunnel the response, especially the Encr-Src header field value. This ensures that intermediaries cannot change the message because nat-sec-proxy can verify the integrity of the message.

F3 200 OK Aloha -> sec-proxy

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP sec-proxy@ipv4.com;branch=z9hG4bKowboy1
Via: SIP/2.0/TCP nat-sec-proxy@ipv6.com;branch=z9hG4bKnappe1
To: <sip:aloha@ipv4.com>; tag=234ff234
From: <sip:nat-sec-proxy@ipv6.com>; tag=990099
Accept: application/sdp, multipart/*
Call-ID: rs8943jdf804s
Contact: <sip: sec-proxy@ipv4.com>
CSeq: 1 OPTIONS
Date: Thu, 14 Aug 2003 13:23:13 GMT
Encr-Src: sec-proxy.ipv4.com
Supported: Encr-Src
Content-Type: multipart/signed;
        protocol="application/pkcs7-signature";
        micalg=sha1; boundary=outer-boundary

--outer-boundary
Content-Type: message/sip

SIP/2.0 200 OK
To: <sip:aloha@ipv4.com>; tag=234ff234
From: <sip:nat-sec-proxy@ipv6.com>; tag=990099
Accept: application/sdp, multipart/*
Call-ID: rs8943jdf804s
Contact: <sip:sec-proxy@ipv4.com>
CSeq: 1 OPTIONS
Date: Thu, 14 Aug 2003 13:23:13 GMT
Encr-Src: sec-proxy.ipv4.com
Supported: Encr-Src
```

--outer-boundary
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--outer-boundary-

After receiving the response, sec-proxy will remove Encr-Src field in the "outer" message in order to make the response routable in the Internet. Afterwards, it forwards the response to nat-sec-proxy.

Exception Handle on nat-sec-proxy

If nat-sec-proxy receives a 420 Bad Extension response, it will create a 420 response to Ying. In this case, Ying cannot establish a connection with Aloha using this extension.

If nat-sec-proxy receives a 200 OK from the receiver, it should first check the integrity of the message. If the identity of the callee cannot be determined because of a problem with the certificate, the security proxy will respond to its UA with a new error code 610 Potential Security Breach specified in Section 6.3.2. Moreover, the security proxy should indicate the reason through a cert-warning in the Encr-Src header field. UA1, which receives this 610 response, must render an appropriate warning to Ying.

Ying may decide to setup the session regardless of the problems with the certificates. Then the UA1 will start with another INVITE again, explicitly setting the optional encr-flag of the Encr-Src header field as the value "offensive". It indicates that broken certificates shall be ignored.

F4. OPTIONS (nat-sec-proxy -> sec-proxy)

If the certificate is valid, and UA2 supports the Encr-Src extension, and the Encr-Src header field is contained in the signed "message/sip" MIME type, nat-sec-proxy knows that the receiver is supporting the extension and located behind its security proxy, sec-proxy. Nat-sec-proxy will look up its internal keyring to find the public key of sec-proxy.

If the public key of sec-proxy is unknown, an OPTIONS request must be needed to obtain the certificate of sec-proxy. Nat-sec-proxy should send a signed OPTIONS request containing itself certificate to receive the certificate of sec-proxy. The certificate is contained in the CMS of the "multipart/signed" message body.

F5. 200 OK (sec-proxy -> nat-sec-proxy)

Sec-proxy receives the OPTIONS that is signed by nat-sec-proxy. It must check the certificate for validity. As UA2 already inspected the certificate, the certificate should be ok. Sec-proxy should respond a signed 200 OK containing its server certificate in the CMS SignedData body. Also it should indicate that it supports the Encr-Src extension.

Note that, oftentimes, a UA knows that its target is capable to answer a secure call and has its security proxy. In this case, the messages F2 and F3 can be skipped. Indeed, if the UA also holds the certificate of the security proxy in the receiving domain, discovering the certificate is not necessary, which means the messages F4 and F5 can be omitted.

F6. INVITE (nat-sec-proxy -> sec-proxy)

Similar to the process described in "Exception Handle on nat-sec-proxy", nat-sec-proxy check the certificate in the received response. If the certificate poses a problem, the security proxy must indicate UA1 with a 610 Potential Security Breach response.

If the certificate held is valid or Ying has explicitly accepted the certificate, nat-sec-proxy will start encrypting SDP on behalf of Ying.

First, nat-sec-proxy must remove Via header fields from the INVITE received from the message F1 for privacy reasons. This header field will be restored when the response of the INVITE returns nat-sec-proxy. Nat-sec-proxy adds a Record-Route header field as it desires to get all the messages that belong to this session. Moreover, it must remove Encr-Src field, and the Contact field must not show the secret information of UA1. After then, it invokes SIP-ALG to translate IPv6 addresses into IPv4 addresses in the message.

A NAT-PT address bind will look like: [fec0::2] <--> 20.0.0.3. And the address table in SIP-ALG will be:

SIP_v6_source: [fec0::2] SIP_v6_dest: [abcd::30.0.0.2]

SIP_v4_source: 20.0.0.3 SIP_v4_dest: 30.0.0.2

After address translation, nat-sec-proxy must encrypt the SDP with the public key of sec-proxy. Then it must sign the encrypted SDP. This encrypted "multipart/signed" body and the "multipart/signed" body containing the tunneled SIP header fields from F1 are packed into a "multipart/mixed" message.

Finally, the request will be assembled by NAT-PT module to send on the Internet.

F6. INVITE (nat-sec-proxy -> sec-proxy)

```
INVITE sip:aloha@ipv4.com SIP/2.0
Via: SIP/2.0/TCP nat-sec-proxy.ipv6.com;branch=z9hG4bKnappe3
To: <sip:aloha@ipv4.com>
From: <sip:ying@ipv6.com>; tag=456222
Call-ID: df49df90s8d
Contact: <sip:ying@nat-sec-proxy.ipv6.com>
CSeq: 1 INVITE
Date: Thu, 14 Aug 2003 13:23:13 GMT
Record-Route: <sip:nat-sec-proxy.ipv6.com;lr>
Content-Type: multipart/mixed; boundary=outer-boundary

--outer-boundary
Content-Type: multipart/signed;
        protocol="application/pkcs7-signature";
        micalg=sha1; boundary=SDP-boundary

--SDP-boundary
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
    name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m;
```

handling=required

****
* Content-Type: application/sdp
*
* v=0
* o=Ying 2890844111 2890844222 IN IP4 20.0.0.3
* s=Session SDP
* c=IN IP4 20.0.0.3
* t=0 0
* m=audio 27002 RTP/AVP 0
* a=rtpmap:0 PCMU/8000
*
****

--SDP-boundary
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--SDP-boundary--

--outer-boundary
Content-Type: multipart/signed;
protocol="application/pkcs7-signature";
micalg=sha1; boundary=header-boundary

--header-boundary
Content-Type: message/sip

Date: Thu, 14 Aug 2003 13:23:13 GMT
Encr-Src: nat-sec-proxy.ipv6.com

--header-boundary

Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--header-boundary--

--outer-boundary--

The text boxed in asterisks ("*") is encrypted by nat-sec-proxy for sec-proxy.

F7. INVITE sec-proxy -> Aloha

When sec-proxy receives the request carrying an encrypted and signed SDP, it decrypts the SDP of the request and substitutes the encrypted SDP with a plain text version. It must add an Encr-Src header field to indicate the server name of the security proxy that encrypted the request, nat-sec-proxy in this example. If the certificate of nat-sec-proxy poses a problem, sec-proxy will add an appropriate cert-warning to the Encr-Src header field. In addition, the security proxy adds a Record-Route header field in order to get all the messages that belong to this session.

Finally, it should forward the decrypted INVITE over a secure connection to UA2.

F7. INVITE (sec-proxy2 -> Aloha)

INVITE sip:aloha@ipv4.com SIP/2.0
Via: SIP/2.0/TLS sec-proxy.ipv4.com;branch=z9hG4bKugel3
Via: SIP/2.0/TCP nat-sec-proxy.ipv6.com;branch=z9hG4bKnappe3
To: <sip:aloha@ipv4.com>

From: <sip:ying@ipv6.com>; tag=456222
Call-ID: df49df90s8d
Contact: <sip:ying@nat-sec-proxy.ipv6.com>
CSeq: 1 INVITE
Date: Thu, 14 Aug 2003 13:23:13 GMT
Encr-Src: nat-sec-proxy.ipv6.com
Record-Route: <sip:sec-proxy.ipv4.com;lr>
Record-Route: <sip:nat-sec-proxy.ipv6.com;lr>
Content-Type: multipart/mixed; boundary=outer-boundary

--outer-boundary
Content-Type: application/sdp

v=0
o=Ying 2890844111 2890844222 IN IP4 20.0.0.3
s=Session SDP
c=IN IP4 20.0.0.3
t=0 0
m=audio 22022 RTP/AVP 0
a=rtpmap:0 PCMU/8000

--outer-boundary
Content-Type: multipart/signed;
protocol="application/pkcs7-signature";
micalg=sha1; boundary=header-boundary

--header-boundary
Content-Type: message/sip
Date: Thu, 14 Aug 2003 13:23:13 GMT
Encr-Src: nat-sec-proxy.ipv6.com

--header-boundary
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--header-boundary--

--outer-boundary--

F8. 200 OK (Aloha -> sec-proxy)

After receiving the INVITE, UA2 first will check potential cert-warning in the Encr-Src header field. Then it will render an appropriate warning to the user if there is a certificate problem.

The INVITE contains the certificate of UA1, as UA1 signed at least the Encr-Src and the Date header fields. UA2 must check the certificate and render appropriate warnings to the user if necessary. UA2 also verifies the integrity of the inner tunneled header fields with the corresponding outer ones.

If the user accepts the call, UA2 sends a 200 OK with its SDP to sec-proxy. An Encr-Src header field containing the server name of sec-proxy must be added to authorize the security proxy encrypt SDP on behalf of UA2. Also the Encr-Src and Data header fields must be signed (tunneled) by UA2.

F8. 200 OK (Aloha -> sec-proxy)

SIP/2.0 200 OK
Via: SIP/2.0/TLS sec-proxy.ipv4.com;branch=z9hG4bKugel3
Via: SIP/2.0/TCP nat-sec-proxy.ipv6.com;branch=z9hG4bKnappe3
To: <sip:aloha@ipv4.com>; tag=2344hz2
From: <sip:ying@ipv6.com>; tag=456222
Call-ID: df49df90s8d
Contact: <sip:aloha@30.0.0.2>
CSeq: 1 INVITE
Date: Thu, 14 Aug 2003 13:23:13 GMT

Encr-Src: sec-proxy.ipv4.com
Record-Route: <sip:sec-proxy.ipv4.com;lr>
Record-Route: <sip:nat-sec-proxy.ipv6.com;lr>
Route: <sip:sec-proxy.ipv4.com;lr>,<sip:nat-sec-proxy.ipv6.com;lr>
Content-Type: multipart/mixed; boundary=outer-boundary

--outer-boundary
Content-Type: application/sdp

v=0
o=Aloha 2890844112 2890844223 IN IP4 30.0.0.2
s=Session SDP
c=IN IP4 30.0.0.2
t=0 0
m=audio 44004 RTP/AVP 0
a=rtpmap:0 PCMU/8000

--outer-boundary
Content-Type: multipart/signed;
protocol="application/pkcs7-signature";
micalg=sha1; boundary=header-boundary

--header-boundary
Content-Type: message/sip

Date: Thu, 14 Aug 2003 13:23:13 GMT
Encr-Src: sec-proxy.ipv4.com

--header-boundary
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--header-boundary-

--outer-boundary-

F9. 200 OK (sec-proxy -> nat-sec-proxy)

Sec-proxy must remove Encr-Src header field and modify the Contact field to hide the sensitive information of UA2. After modification, it encrypts SDP using the public key of nat-sec-proxy. A "multipart/mixed" MIME body will be created finally.

F9. 200 OK (sec-proxy -> nat-sec-proxy)

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP nat-sec-proxy.ipv6.com;branch=z9hG4bKnappe3
To: <sip:aloha@ipv4.com>; tag=2344hz2
From: <sip:ying@ipv6.com>; tag=456222
Call-ID: df49df90s8d
Contact: <sip:aloha@sec-proxy.ipv4.com>
CSeq: 1 INVITE
Date: Thu, 14 Aug 2003 13:23:13 GMT
Record-Route: <sip:sec-proxy.ipv4.com;lr>
Record-Route: <sip:nat-sec-proxy.ipv6.com;lr>
Route: <sip:nat-sec-proxy.ipv6.com;lr>
Content-Type: multipart/mixed; boundary=outer-boundary
Content-Length: ...

--outer-boundary
Content-Type: multipart/signed;
            protocol="application/pkcs7-signature";
            micalg=sha1; boundary=SDP-boundary

--SDP-boundary
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
      name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m;
      handling=required
Content-Length=...

**************************************************
```

* Content-Type: application/sdp
*
* v=0
* o=Aloha 2890844112 2890844223 IN IP4 30.0.0.2
* s=Session SDP
* c=IN IP4 30.0.0.2
* t=0 0
* m=audio 27808 RTP/AVP 0
* a=rtpmap:0 PCMU/8000
****

--SDP-boundary
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--SDP-boundary--

--outer-boundary
Content-Type: multipart/signed;
        protocol="application/pkcs7-signature";
        micalg=sha1; boundary=header-boundary

--header-boundary
Content-Type: message/sip

Date: Thu, 14 Aug 2003 13:23:13 GMT
Encr-Src: sec-proxy.ipv4.com

--header-boundary
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--header-boundary--

--outer-boundary--

F10. 200 OK (nat-sec-proxy -> Ying)

Nat-sec-proxy receives the signed and encrypted response from sec-proxy. It decrypts the SDP of the response and substitutes the encrypted SDP with a plain text version. Also, it adds an Encr-Src header field whose value is the server name from the certificate of sec-proxy. Additionally, nat-sec-proxy must restore the Via header field that has been removed from F1.INVITE. Finally, it requires IP address translation with SIP-ALG.

Nat-sec-proxy forwards the response to Ying in a secure manner.

F10. 200 OK (nat-sec-proxy -> Ying)

SIP/2.0 200 OK
Via: SIP/2.0/TLS [fec0::2];branch=z9hG4bKnas12f8
To: <sip:aloha@ipv4.com>; tag=2344hz2
From: <sip:ying@ipv6.com>; tag=456222
Call-ID: df49df90s8d
Contact: <sip:aloha@sec-proxy.ipv4.com>
CSeq: 1 INVITE
Date: Thu, 14 Aug 2003 13:23:13 GMT
Encr-Src: sec-proxy.ipv4.com
Record-Route: <sip:sec-proxy.ipv4.com;lr>
Record-Route: <sip:nat-sec-proxy.ipv6.com;lr>
Content-Type: multipart/mixed; boundary=outer-boundary

--outer-boundary
Content-Type: application/sdp

v=0
o=Aloha 2890844112 2890844223 IN IP6 [abcd::30.0.0.2]
s=Session SDP
c=IN IP6 [abcd::30.0.0.2]
t=0 0
m=audio 25880 RTP/AVP 0
a=rtpmap:0 PCMU/8000

--outer-boundary
Content-Type: multipart/signed;
protocol="application/pkcs7-signature";
micalg=sha1; boundary=header-boundary

--header-boundary
Content-Type: message/sip

Date: Thu, 14 Aug 2003 13:23:13 GMT
Encr-Src: sec-proxy.ipv4.com

--header-boundary
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--header-boundary--

--outer-boundary--

F11. ACK (Ying -> nat-sec-proxy)

UA1 will first check cert-warning in the Encr-Src header field. Then it will render an appropriate warning to the user if there is a certificate problem.

The response contains the certificate of UA2, as UA2 signed at least the Encr-Src and the Date header fields. UA2 must check the certificate and render appropriate warnings to the user if necessary. UA2 also verifies the integrity of the inner tunneled header fields with the corresponding outer ones.

If the user does not want to proceed with the call because of a rendered warning (cert-warning, broken header field integrity), the UA will terminate the call by sending an ACK request followed by a BYE.

If the user accepts the call, UA1 sends an ACK request to UA2. The Encr-Src header field is not needed in ACK message because no SDP packets are carried by an ACK request and are needed to be encrypted by nat-sec-proxy as well. Note that the ACK must not be sent end-to-end encrypted, because the security proxies are stateful and no response will be created for ACK.

F11. ACK (Ying -> nat-sec-proxy)

```
ACK sip:aloha@ipv4.com SIP/2.0
Via: SIP/2.0/TLS [fec0::2];branch=z9hG4bKnockOut1
To: <sip:aloha@ipv4.com>; tag=2344hz2
From: <sip:ying@ipv6.com>; tag=456222
Call-ID: df49df90s8d
CSeq: 1 ACK
Date: Thu, 14 Aug 2003 13:23:13 GMT
Route: <sip:nat-sec-proxy.ipv6.com;lr>,<sip:sec-proxy.ipv4.com;lr>
Content-Length: 0
```

F12. ACK (nat-sec-proxy -> sec-proxy)

Nat-sec-proxy hides the Ying's private information and performs address translation with NAT-PT box.

F12. ACK (nat-sec-proxy -> sec-proxy)

ACK sip:aloha@ipv4.com SIP/2.0
Via: SIP/2.0/TCP nat-sec-proxy.ipv6.com;branch=z9hG4bKumKum1
To: <sip:aloha@ipv4.com>; tag=2344hz2
From: <sip:ying@ipv6.com>; tag=456222
Call-ID: df49df90s8d
CSeq: 1 ACK
Date: Thu, 14 Aug 2003 13:23:13 GMT
Route: <sip:sec-proxy.ipv4.com;lr>
Content-Length: 0

F13 ACK (sec-proxy -> Aloha)

Sec-proxy relays the ACK to Aloha.

F13 ACK (sec-proxy -> Aloha)
ACK sip:aloha@ipv4.com SIP/2.0
Via: SIP/2.0/TLS sec-proxy.ipv4.com;branch=z9hG4bKugel21
Via: SIP/2.0/TCP nat-sec-proxy.ipv6.com;branch=z9hG4bKumKum21
To: <sip:aloha@ipv4.com>; tag=2344hz2
From: <sip:ying@ipv6.com>; tag=456222
Call-ID: df49df90s8d
CSeq: 1 ACK
Date: Thu, 14 Aug 2003 13:23:13 GMT
Content-Length: 0

Up to now, an end-to-end secure SIP session has been established from an IPv6 UA to an IPv4 UA. For sessions in the opposite direction, the procedures of call establishment are similar to those described above.