SIP on an Overlay Network

XIAO WU

KTH Information and Communication Technology

Master of Science Thesis
Stockholm, Sweden 2009

TRITA-ICT-EX-2009:105

SIP on an Overlay Network

Xiao Wu

14 September 2009

Academic Supervisor and Examiner: Gerald Q. Maguire Jr.
Industrial supervisor: Jorgen Steijer, Opticall AB
School of Information and Communication Technology
Royal Institute of Technology (KTH)
Stockholm, Sweden

Abstract

With the development of mobile (specifically: wide area cellular telephony) technology, users' requirements have changed from the basic voice service based on circuit switch technology to a desire for high speed packet based data transmission services. Voice over IP (VoIP), a packet based service, is gaining increasing attention due to its high performance and low cost. However, VoIP does not work well in every situation. Today Network address translation (NAT) traversal has become the main obstruction for future VoIP deployment.

In this thesis we analyze and compare the existing NAT traversal solutions. Following this, we introduce a VoIP over IPSec (VOIPSec) solution (i.e., a VoIP over IPSec virtual private network (VPN) scheme) and an extended VOIPSec solution mechanism. These two solutions were tested and compared to measure their performance in comparison to a version of the same Session Initiation Protocol (SIP) user agent running without IPSec.

In the proposed VOIPSec solution, the IPSec VPN tunnel connects each of the SIP clients to a SIP server, thus making all of the potential SIP participants reachable, i.e., solving the NAT traversal problem. All SIP signaling and media traffic for VoIP calls are transmitted through this prior established tunnel. This VPN tunnel provides the desired universal means for VoIP traffic to traverse NAT equipment. Additionally, the IPSec VPN also guarantees the security of VoIP calls at the IP level.

In order to improve the security level of media streams for the VOIPSec solution, we deployed and evaluated an extended VOIPSec solution which provides end-to-end protection of the real time media traffic. In this extended VOIPSec solution, we used SRTP instead of RTP to carry the media content. This extended method was shown to provide all of the advantages of VOIPSec and SRTP without any additional delay for the media traffic (as compared to the VoIPSec solution).

Note that the solution proposed in this thesis may be of limited practical importance in the future as more NATs become VoIP capable; but the solution is currently essential for facilitating the increasing deployment of VoIP systems in practice. For VoIP calls that do **not** need end-to-end security, we recommend the use of the VOIPSec solution as a means to solve the NAT traversal problem and to protect traffic at the IP level. When application to application security is **not** needed we prefer the VOIPSec solution to the extended VOIPSec solution for the following reasons: (1) our test results show that the time for call setup for the extended VOIPSec solution is twice time the time needed for the VOIPSec solution and the extended VOIPSec solution requires the use of user agents that support SRTP. While, the VOIPSec solution does not require a special user agent and all VoIP clients in the market are compatible with this solution. However, when more SIP user agents add support for SRTP, the extended VOIPSec solution will be applicable for users of these SIP user agents.

Sammanfattning

Med utvecklingen av mobil (specifikt: wide area cellulär telefoni)-teknik, har användarnas krav ändras från den grundläggande röst-tjänst som bygger på krets kopplad teknik till att vilja ha hög-hastighets paket baserade dataöverföringstjänster. Voice over IP (VoIP) som vinner allt mer uppmärksamhet på grund av sin höga prestanda och låga kostnader är en paket baserad telefon tjänst. Däremot fungerar VoIP inte bra i alla situationer. Network address translation (NAT) har blivit det största hinder för en framtida användning av VoIP.

I denna avhandling analyserar vi och jämför nuvarande NAT lösningar. Efter detta inför vi en VoIP över IPSec (VOIPSec) lösning (dvs. ett VoIP över IPSec Virtual Private Network (VPN) system) och en utvidgad VOIPSec lösnings mekanism. Dessa två lösningar testas och jämfördes för att mäta prestationer i förhållande till en version av samma SIP User Agent som körs utan IPSec.

I den föreslagna lösningen VOIPSec ansluter IPSec en VPN-tunnel till varje SIP-klient och SIP-server, vilket gör att alla de potentiella SIP deltagarna kan nås, dvs eventuella NAT problem löses. All SIP-signalering och media trafik för VoIP-samtal överförs via denna etablerade tunnel. Denna VPN-tunnel ger allmänna medel för VoIP-trafik att passera NAT utrustningen. Dessutom ger IPSec VPN också garanterad säkerheten för VoIP-samtal på IP-nivå.

För att förbättra skyddsnivån för mediaströmmar med VOIPSec, skapade vi och utvärderade en utsträckt VOIPSec lösning som innehåller end-to-end skydd av realtids media trafik. I denna utökade VOIPSec lösning, använde vi SRTP stället för RTP för att bära medieinnehåll. Denna utvidgade metod visade sig ge alla fördelar VOIPSec och SRTP kunde erbjuda utan ytterligare dröjsmål för media trafiken (jämfört med VoIPSec lösningen).

Observera att den lösning som föreslås i denna avhandling kan vara av begränsad praktisk betydelse i framtiden då fler NAT lösningar blir VoIP kapabla, men lösningen är idag nödvändigt för att underlätta den ökande användningen av VoIP-system i praktiken. För VoIP-samtal som inte behöver end to end säkerhet rekommenderar vi användning av VOIPSec lösningen som ett sätt att lösa NAT problem och för att skydda trafiken på IP-nivå. När end to end säkerhet inte behövs föredrar vi VOIPSec lösningen av följande skäl: (1) våra testresultat visar att tiden för samtal inställning för det förlängda VOIPSec lösningen är dubbelt den tid som krävs för VOIPSec lösningen och den utökade VOIPSec lösningen kräver användning av användarprogram som stödjer SRTP. Medan VOIPSec lösningen inte kräver en speciell användar agent och alla VoIP-klienter på marknaden är kompatibla med denna lösning. Men när fler SIP användaragenter får stöd för SRTP, kommer den förlängda VOIPSec lösning tillämpas för användare av dessa SIP användarprogram.

Table of contents

List of Figures

List of Tables

List of Acronyms

AH	Authentication protocol
ALG	Application Layer Gateway
CODEC	Coder/Decoder
DH	Diffie-Hellman
DOI	Domain of Interpretation
ESP	Encapsulating Security Payload
ICE	Interactive Connectivity Establishment
IKE	Internet Key Exchange
IPSec	Internet Protocol Security
ISAKMP	Internet Security Association and Key Management Protocol
ISP	Internet Service Provider
MIKEY	Multimedia Internet KEYing
MKI	Master Key Identifier
NAT	Network Address Translation
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
SA	Security Association
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SPI	Security Parameter Index
SRTP	Secure Real-time Transport Protocol
SSL	Secure Sockets Layer
STUN	Session Traversal Utilities for NAT
TLS	Transport Layer Security
TURN	Traversal Using Relay NAT
UA	User Agents
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
VoIP	Voice over IP
VOIPSec	VoIP works with IPSec
VPN	Virtual Private Network
σ	Standard Deviation – Symbolized by a lower case Greek sigma

1 Introduction

1.1 General Overview

In the telephony world, digital circuit switched networks replaced analog circuit-switched telephone networks a couple of decades ago.[1] Today, the public switched telephone network (PSTN) is a digital circuit-switched telephone network interconnecting public telephony networks around the world. This PSTN offers reliability, good voice quality, minimal delay, and worldwide phone connectivity. The PSTN's characteristics are well understood for voice communication and low speed data transmission (i.e., sending encoded data traffic across the PSTN using modems).

With the development of packet switched technology, users' requirements have changed from basic voice service based on circuit switched technology to high speed packet based data transmission services. Voice over IP (VoIP) technology, which is based on packet switched technology, is gaining increasing attention for its efficiency and low cost for long distance communications.[2] VoIP vendors point out that VoIP rides a new wave of changes as the telecommunications industry moves from circuit switched networks to packet switched networks. However, some analysts argue that it will be a long time before corporations abandon proven private branch exchange (PBX) systems and use packet-based networks for data, voice, and video.[3]

Integrating a packet switched network with a circuit switched network is necessary in order to realize and offer the potential of saving significant cost, gaining effective performance, and improving interconnectivity with mobile terminals. It should be noted that third generation cellular infrastructures are already in the process of eliminating circuit-switched voice and this trend is likely to continue. Thus, except for legacy PSTN systems, there will be fewer and fewer circuit-switched systems that a VoIP system needs to interconnect to.

The voice signal in VoIP is segmented into frames, encoded, and encapsulated in RTP (see section 2.3). The packets are then transported over an IP network. A number of VoIP protocols exist, including: H.323, SIP (see section 2.1), Media Gateway Control Protocol (MGCP), T.38, etc.[4] This thesis will focus on the use of the Session Initiation Protocol (SIP).

1.2 Problem Statement

Although VoIP has gained popularity in both consumer and business markets in recent years, there is one major challenge which influences the adoption of VoIP technology. VoIP does **not** work well in every situation, especially when the terminals are behind a Network Address Translation (NAT) device. This is because the NAT creates a private network, thus devices inside this private network can use private IP addresses. Such private IP addresses are **not** accessible from the global Internet.

In short, the main problems caused by NAT for a VoIP call occurs when the device's private IP address(es) are encoded into the message header and Session Description Protocol (SDP) bodies of SIP packets. Unfortunately in most cases, the private IP addresses contained in the SDP are not processed by the NAT device, as most NAT devices do not provide application specific processing for SIP packets as they traverse the NAT. Thus although the IP addresses in the outer packet headers are

correctly translated by the NAT from the private to the public IP address space, the private IP addresses within these packets are not translated. This causes the destination to be unable to respond, as it can not send RTP packets to the source as these are *non-routable* private IP addresses[5].

In this paper, we will analyze how to solve the NAT traversal problem using Virtual Private Network (VPN) technology. The VPN mechanism not only can be used to establish a tunnel to traverse a NAT or a firewall, but this tunnel also secures the voice and data communication.

Increasingly providing privacy and authentication of voice and other data traffic is an essential requirement for telecommunication services. However, the major reason that we considered the use of VPN technology is because a large fraction of the NATs that have been sold are capable of properly handling VPNs - i.e., they feature application layer gateway functions for VPNs, but not (yet) for VoIP. Note that the solution proposed in this thesis may be of limited practical importance in the future as more NATs become VoIP capable; but the solution is currently essential for facilitating the increasing deployment of VoIP systems in practice.

2 VoIP Technology Overview

Voice over IP technology is gaining more and more attention for its efficiency and low cost. In this chapter we introduce the basic elements of a VoIP system. The SIP and RTP protocols are described briefly, with references to additional information about these protocols.

2.1 Session Initiation Protocol

Session Initiation Protocol (SIP) is a protocol developed by the Internet Engineering Task Force (IETF) to assist in providing advanced telephony services across the Internet. It is used for negotiating the parameters for establishing, modifying, and terminating a SIP session.[6]

2.1.1 SIP Network Elements

The SIP architecture defines four major components: SIP User Agents, SIP Registrar Servers, SIP Proxy Servers, and SIP Redirect Servers. These components differ in their logical functions. To increase the speed of processing and make it simpler to configure, SIP register server, SIP proxy server, and SIP redirect server often are co-located on a single computer; this computer is generally referred to as a SIP server.

SIP User Agents (UAs)[7] are the endpoint devices in a SIP network. They can be further divided into two components: User Agent Client (UAC) and User Agent Server (UAS). The UAC initiates SIP requests to a SIP server in order to establish a SIP session. The UAS responds to requests it has received.

SIP Registrar Servers[7] accept REGISTER requests from UAs and maintain information about their location. SIP Proxy Servers[7] are an intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other clients. SIP Redirect Servers[7] are user agent servers that generate 3xx responses to requests they receive, directing the clients to contact an alternate set of URIs.

2.1.2 SIP Messages

SIP messages are sent between SIP elements to establish, manipulate, and terminate the SIP session. If UDP is used to transport the SIP message, then each message is transported in a separate UDP datagram. As usual each IP packet is routed independent by the network. SIP messages are either requests from the server to the client or responses to a request. The general format of all the messages consists of a start-line, one or more header fields, an empty line, and an optional message body.

The basic requests are INVITE, ACK, BYE, OPTIONS, CANCEL, and REGISTER. The responses are of the form: 1xx, 2xx, 3xx, 4xx, 5xx, and 6xx, where the first digit of the response indicates the class of the response and the remaining digits indicate the particulars of the response. The purposes and meanings for each Request and Response are shown in Table 2-1[8] and Table 2-2.[8]

Table 2-1. SIP Requests

Request	Purpose
INVITE	Invites a user to join a call.
ACK	Confirms that a client has received a final response to an INVITE
BYE	Terminates the call between two of the users on a call
OPTIONS	Requests information on the capabilities of a server
CANCEL	Ends a pending request, but does not end the call.
REGISTER	Provides the map for address resolution; this lets a server know the location of a user.

Table 2-2. SIP Responses

Response	Meaning
1xx	Informational or Provisional - request received, continuing to process the request
2xx	Final - the action was successfully received, understood, and accepted
3xx	Redirection - further action needs to be taken in order to complete the request
4xx	Client Error - the request contains bad syntax or cannot be fulfilled at this server
5xx	Server Error - server failed to fulfill an apparently valid request (Try another server!)
6xx	Global Failure - the request cannot be fulfilled at any server (Give up!)

2.1.3 SIP Flows

2.1.3.1 Registration

Each SIP UA registers its location with a REGISTRAR server when it connects to the SIP system. The SIP UA sends a REGISTER message which contains its current location information. The message flows between servers and SIP UA in a example registration process are shown in Figure 2-1.[9]

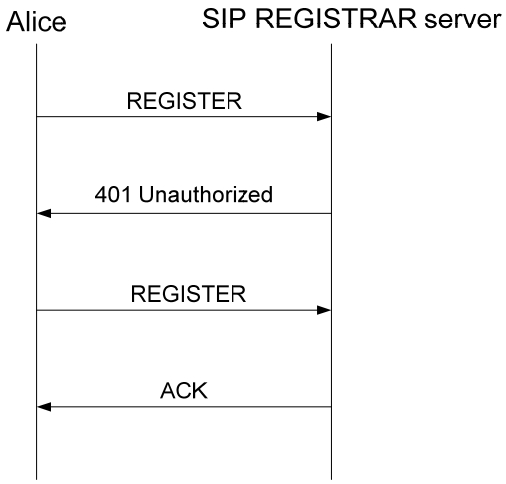


Figure 2-1. SIP Registration Process

Alice sends an SIP REGISTER request to the SIP server. It includes Alice's contact list. The SIP server provides a challenge to Alice and sends a 401 Unauthorized response back. Alice encrypts the user information (valid user ID and password) according to the challenge which is issued by the SIP server and sends it with a new REGISTER message to the SIP server. After successful user verification, the SIP server registers the user in its contact database and sends a 200 OK response to Alice.[9]

An example of an SIP REGISTER message is shown below.

```
REGISTER sip:registrar.opticall.com SIP/2.0
Via: SIP/2.0/UDP alicespc.opticall.com:5060;branch=random
From: alice <sip:alice@opticall.com>;tag= random
To: alice <sip:alice@opticall.com>
Contact: "alice"<sip:alice@217.75.104.150>
Call-ID: random@opticall.com
CSeq: 796 REGISTER
Expires: 1800
Max-Forwards: 70
Content-Length: 0
```

2.1.3.2 SIP Session Setup

A typical SIP session setup is shown in Figure 2-2. In this example, two SIP UAs complete a successful call using two proxy servers. Proxy 1 is the default outbound proxy for Alice. Proxy 2 is the default inbound proxy for Bob. The first INVITE request is sent by the caller (Alice) to initiate the call with the callee (Bob). A 407 Proxy Authorization response containing challenge information is sent back from Proxy 1 to Alice, because the first INVITE request Alice sent to Proxy 1 does not contain the Authorization credentials that Proxy 1 requires. Alice sends a new INVITE request which contains the correct credentials (valid user ID and password). A 100 Trying response from the server indicates that the INVITE message is received. The 180 Ringing message provides feedback from the callee to show it has received the INVITE message. As soon as the callee goes off hook (for example, by answering the call), a 200 OK response is sent to the caller. The ACK confirms that the call setup was successful. The INVITE, OK, and ACK provide 3-way-handshaking, to set up a call reliably. Media flows are sent between the two UAs after the SIP session is set up. Media flows utilize the Real-time Transport Protocol (see section 2.3). The SIP session is terminated by a BYE message.[9]

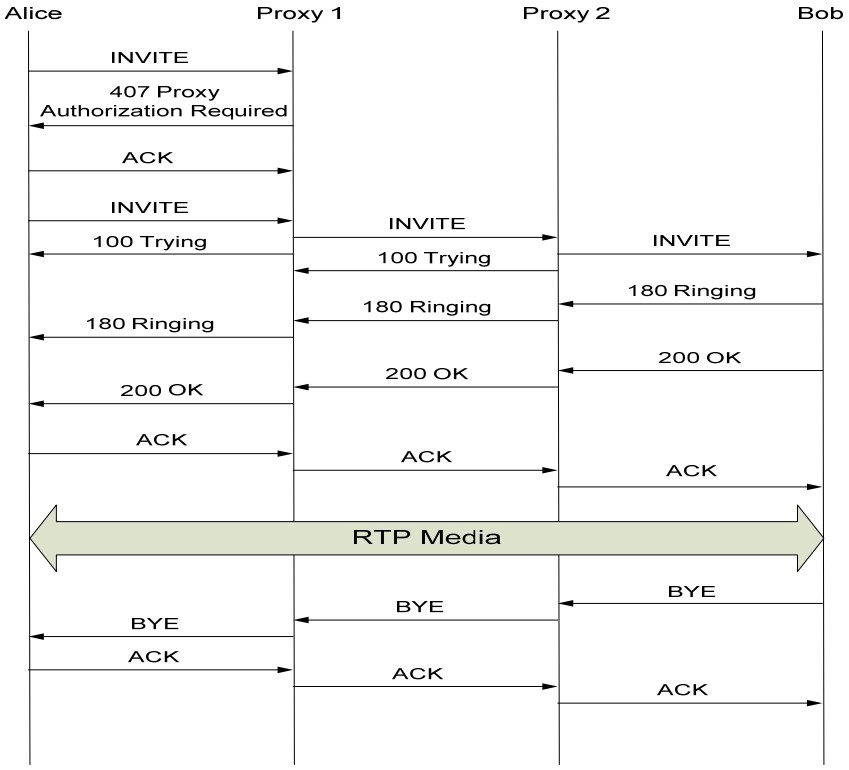


Figure 2-2. SIP session setup process

2.2 Session Description Protocol (SDP)

The Session Description Protocol (SDP)[10] defines a format for describing sessions,. It is intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation. SDP does **not** provide media content, but simply provides a means for two terminals to agree on one or more media types and formats.

In an SIP system, SIP messages carry session descriptions to create an SIP session. This session description, commonly formatted using SDP, is used to negotiate and agree on a set of compatible media types between the participants. A SDP session description includes following elements:[11]

- Session name and purpose
- Time(s) the session is active
- The media comprising the session
- Information necessary to receive those media (IP addresses, ports, formats, and so on)

An example of an SDP session description carried by an SIP message for a session setup is:

```
v=0
o= alice 2614193117 2614193186 IN IP4 217.75.104.150
s=Minisip
c=IN IP4 217.75.104.150
t=0 0
m=audio 8000 RTP/AVP 0 8
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
```

An SDP session description includes several fields which are:[11]

Version number (v)	shows the version of the session description protocol
Origin (o)	indicates the originator of the session (username, address, session identifier, etc.)
Session name (s)	gives the textual session name
Connection information(c)	contains connection data (network type, address type, connection address)
Time (t)	specifies the start and stop time for a session
Media (m)	contains media descriptions included media type, transport port
Attribute (a)	specifies additional properties.

2.3 Real-time Transport Protocol (RTP)

The Real-time Transport Protocol (RTP) defines a standardized packet format for delivering audio, video, timed text, etc. over a transport protocol. RTP together with the RTP Control Protocol (RTCP) typically use the User Datagram Protocol (UDP) as their transport protocol. However, they could utilize other transport protocols, such the stream control transport protocol or the datagram congestion control protocol.

RTCP supplies flow and congestion control information that the end points can use to adjust their sending rate, terminate a session, etc.[12] An RTP application session opens two ports: one for RTP and one for RTCP. RTCP periodically transmits control packets to participants in a multimedia session.

3 NAT Traversal: Problem and Solutions

This chapter presents the NAT mechanism. First, the NAT concept and how it works is introduced in Section 3.1.1. Section 3.1.2 describes four types of NATs and compares the different mechanisms of these NATs. Section 3.2 presents the main problem faced by VoIP applications due to NAT. The chapter ends with a description of four existing solutions that are used to solve the NAT traversal problem.

3.1 Network address translation (NAT)

3.1.1 What is NAT?

Network Address Translation (NAT)[13] is a popular method for expanding the local IPv4 address place. It enables multiple hosts on a private network to access the Internet using a single public IP address.

Figure 3-1 shows the typical use of an NAT. A local network uses one of the designated private IP address subnets, in this case: 192.168.1.0/24. The NAT router has a private address (192.168.1.1) in this private network. The router is connected to the Internet with a public address (213.132.115.12) by an Internet service provider (ISP). This NAT router acts as a gateway between the local private network and the Internet.

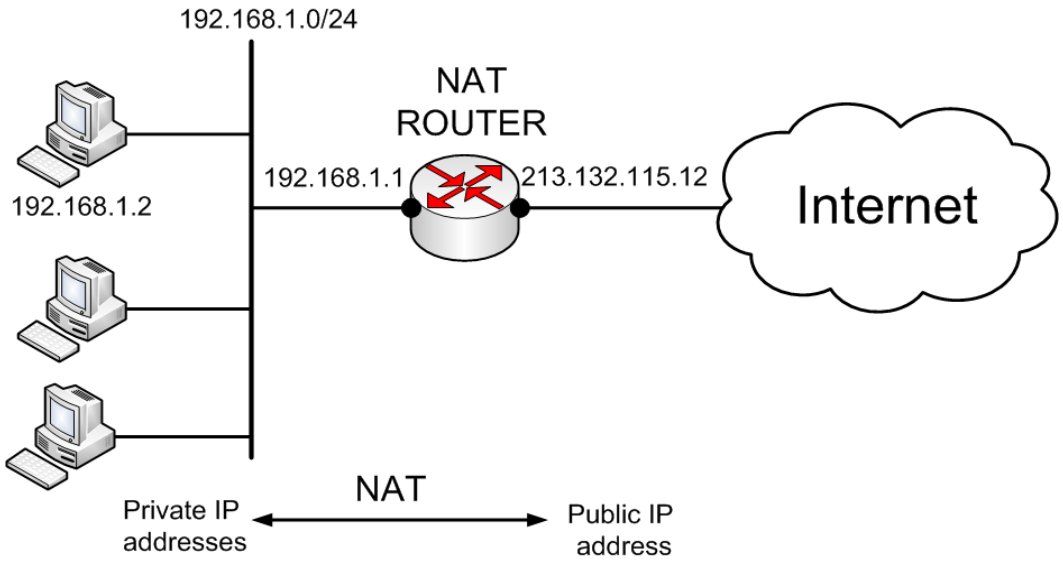


Figure 3-1. NAT Operation

When a client on the internal network, for instance 192.168.1.2, wishes to send packets to a machine on the Internet, it simply sends IP packets to the destination IP address. These packets contain the destination's IP address (in this case, this is a public IP address of the destination), its own source IP address (the private IP address of this client, in this case: 192.168.1.2), a source TCP/UDP/… port, and a destination TCP/UDP/… port.

When the packets pass through the NAT the header of the IP packet will be modified so that the packet appears to be coming from the NAT itself. The NAT records the changes it makes in its translation table so that it can reverse these changes for returning packets. Additionally, as the NAT is

acting as a stateful firewall, it makes an entry in its routing table to ensure that the return packets are passed through the firewall and are not blocked. For example, it might replace the source IP address with its external address (i.e. 213.132.115.12) and replacing the source port with a dynamically assigned port number (this port number is dynamically assigned by the NAT to be used for traffic from this internal host to the destination).

It is important to note that neither the internal machine nor the Internet host is aware of these translation steps. This is simultaneously one of the advantages of NAT and in the case of VoIP it is a source of lots of problems, since the internal machine does **not** know that it is behind a NAT!

3.1.2 NAT Types

NAT implementations can be classified into four classes: full cone NAT, restricted cone NAT, port restricted cone NAT, or symmetric NAT -- based upon the details of how the NAT performs the translation process.[14] After introducing these different types of NATs, we will explain why we need to determine which type of NAT is on the path between a VoIP terminal and the Internet.

3.1.2.1 Full cone NAT

Figure 3-2 shows the structure of a Full cone NAT. In this type of NAT, all requests from the same internal IP address and port (192.168.1.2:21) are mapped to the same external IP address and port (213.132.115.12:12345). Furthermore, **any** external host can send a packet to the internal host, by sending a packet to the mapped external address. This type of NAT is the simplest type of NAT and is rather easy for SIP to deal with - as we only need to determine what the external address and port number are for the client's private address and source port. Once this information is known, then this information can be placed into the SDP message.

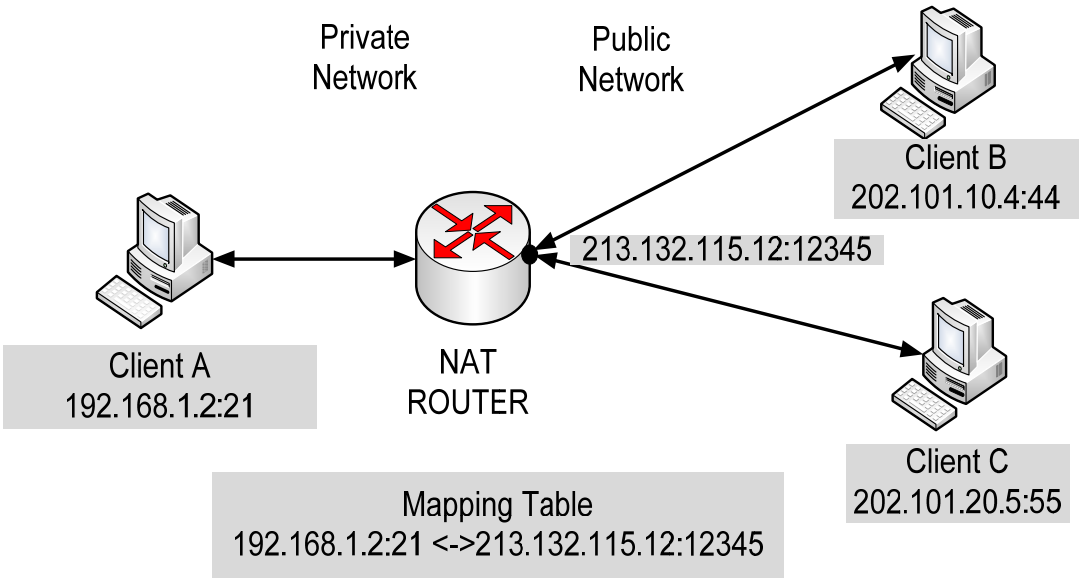


Figure 3-2. A mapping table of a Full Cone NAT

3.1.2.2 Restricted cone NAT

Figure 3-3 shows the structure of a restricted cone NAT: all requests from the same internal IP address and port (192.168.1.2:21) are mapped to the same external IP address and port (213.132.115.12:12345). However, only the external host (Client B) can send a packet to the internal host. Additionally, this external host can only send a packet to this internal host *if the internal host has previously sent a packet to this host*. Unfortunately, this means that this internal host is only available to a host that it has previously sent traffic to — and this earlier traffic has to have been during the time that the address and port mapping is in the mapping table. This limitation is due to the fact that after some period of time the NAT will remove the mapping **unless** the internal host has sent additional traffic to the external host. The time before the NAT garbage collects "unused" mapping entries varies from NAT to NAT, thus an internal host and external host will experience unpredictable problems in communication – as neither knows when the mapping entry for their communication will be removed!

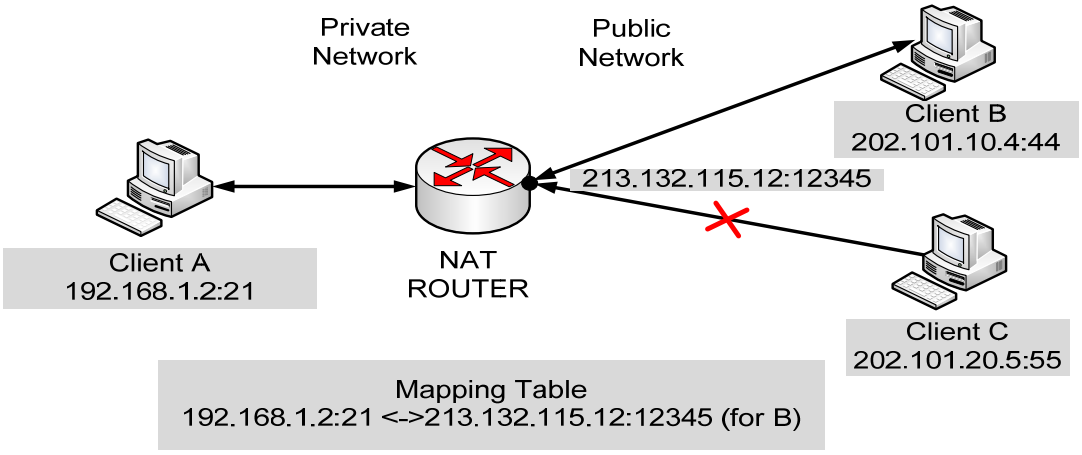


Figure 3-3. A mapping table of Restricted Cone NAT

3.1.2.3 Port restricted cone NAT

A port restricted cone NAT is similar to a restricted cone NAT, but the restriction now includes the external host's port number.

Figure 3-4 shows that an external host (Client B) can send a packet, with its source IP address and a particular source port (202.101.10.4:44) , to the internal host (Client A) , but only if Client A has previously sent a packet to this IP address and source port. If Client B tries to send a packet from its source port 55, to the destination 213.132.115.12:12345, this packet will be blocked.

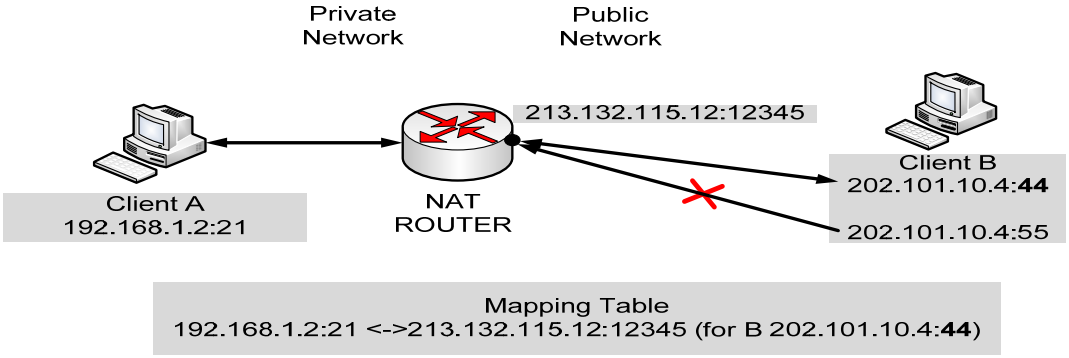


Figure 3-4. A mapping table of Port Restricted Cone NAT

3.1.2.4 Symmetric NAT

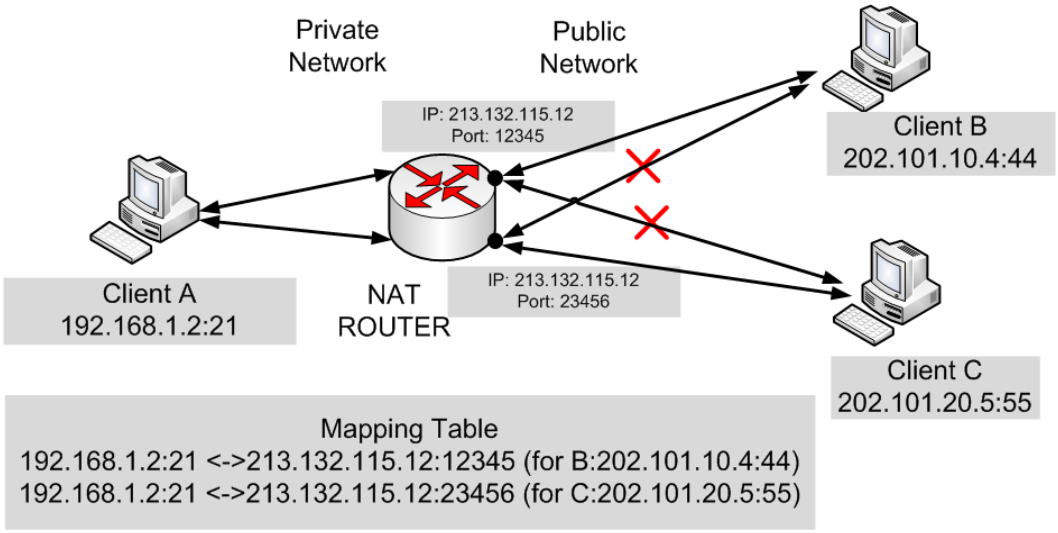


Figure 3-5, is one where all requests from the same internal IP address and port (for example, 192.168.1.2:21) to a specific destination IP address and port (for example, Client B 202.101.10.4:44) are mapped to the same external IP address and port (for example, 213.132.115.12:12345). If the same host sends a packet with the same source address and port, but to a ***different*** destination (for example, Client C 202.101.20.5:55), a ***different*** mapping is used (e.g., 213.132.115.12:67890). Furthermore, only the external host that receives a packet from this address and port combination can send a packet back to the internal host (with this specific address and port combination).

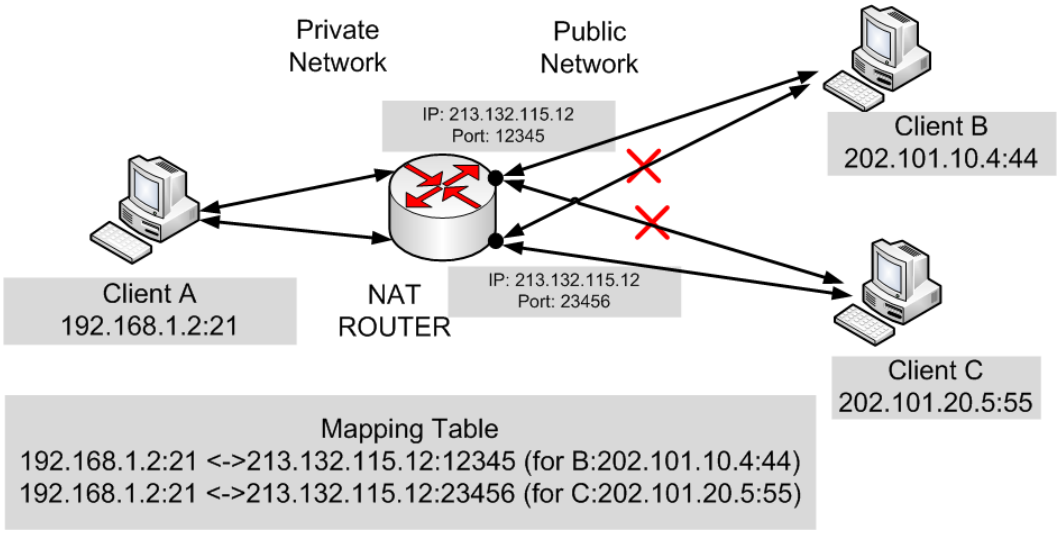


Figure 3-5. A mapping table of Symmetric NAT

3.1.3 The NAT Traversal Problem for SIP

NAT devices are commonly used to reduce the use of IPv4 addresses in modern networks. However, NAT breaks IP end-to-end connectivity as it was originally conceived, which causes problems not only for SIP signaling, but also RTP media transmission.

Before making the SIP call, the SIP session should be established between the caller and the callee. The INVITE message, the first SIP message sent from caller to callee, is used to initiate the SIP session. The example below shows an SIP INVITE message as sent behind NAT.[15]

```
INVITE sip:9002@213.167.88.44;user=phone SIP/2.0
Via: SIP/2.0/UDP 192.168.0.21:5060
From: <sip:9003@213.167.88.44;user=phone>;tag=2577892
To: <sip:9002@213.167.88.44;user=phone>
Call-ID: 2711238610@192.168.0.21
CSeq: 1 INVITE
Contact:  <sip:9003@192.168.0.21:5060;user=phone;transport=udp>
Content-Length: 282
Content-Type: application/sdp
v=0
o=9003 97673 97673 IN IP4 192.168.0.21
s= Minisip
c=IN IP4 192.168.0.21
m=audio 16384 RTP/AVP 0 18 8 101
a=rtpmap:0 PCMU/8000/1
```

The fields of the SIP packet and SDP session description are shown in Table 3-1[16] and Table 3-2.[17]

Table 3-1. Field description of SIP packets

Field	Description
Via	The Via header field indicates the transport used for the transaction and identifies the location where the response is to be sent.
From	The logical sender
To	The logical recipient of the message
Call-ID	The Call-ID header field uniquely identifies a particular invitation or all registrations of a particular client.
CSeq	The Command Sequence Number (CSeq) header field serves as a way to identify and order transactions
Contact	The Contact header field provides a SIP or SIPS URI that can be used to contact that specific instance of the UA for subsequent requests.

Table 3-2. SDP session description

Field	Description
v	protocol version
o	owner/creator and session identifier
s	session name
c	connection information
m	media name and transport address
a	zero or more session attribute lines

The first NAT traversal problem is the **`Via`** Header in INVITE message. When the callee gets the INVITE request and tries to send a response back to the caller, it sends the response back using the address in the **`Via`** header. However, in the example above, the caller is behind an NAT and has a private IP address (192.168.0.21) in the **`Via`** header. However, the response from the callee cannot be routed back to the caller using this address, as it a private address hence it is not globally routable.

The second problem is that the address included in the **`Contact`** header to route future requests is also a private IP address.

The final problem occurs when sending RTP packets back to the originator. The SDP messages are used to negotiate session parameters for RTP media transport, such as media CODEC, IP address, port, etc.). However, because there was a private IP address in the "**c**" field of SDP message and the RTP packets cannot be routed from the public network to the private network the RTP packets from the callee will **not** make it to the caller. However, if the callee is actually at a public IP address and not behind the NAT it will receive RTP packets from the caller (as the NAT can forward UDP packets from the private network to a globally routable IP address)!

3.2 Existing solutions

There are several techniques and solutions to solve these NAT traversal problems. In the following sections, we will present and compare typical NAT traversal schemes for SIP. A SIP user agent may implement zero or more of these techniques.

3.2.1 Session Traversal Utilities for NAT (STUN)

Session Traversal Utilities for NAT (STUN)[18] is used as an NAT traversal method for interactive IP communications. It provides a mechanism for the client to discover whether it is behind an NAT, what the specific the type of NAT is, and what mapping the NAT has allocated for this client's private IP address and port (i.e., what public IP address and port corresponds to this private IP address and port).

STUN is a client-server protocol. In Figure 3-6, the STUN-enabled SIP client sends a request to bind its private IP address and public IP address to an STUN server. The NAT will modify the source transport address and port number of this packet when the binding request message passes through the NAT. After the STUN server receives this packet, it sends a binding response back to the STUN client containing the client's mapped IP address and port on the public side of the NAT. When the packet passes back through the NAT, the NAT will modify the destination address to be the client's private IP address and the client will receive the STUN response. Now the client knows its external public address and port combination (at least in terms of what mapping the NAT did when sending a packet to the STUN server from this client's private IP address and source port). The mapped public IP address and port provided by the STUN server can be used as the value in the "Contact" field in a SIP call establishment message, thus the Contact field contains a valid globally routable IP address and port.

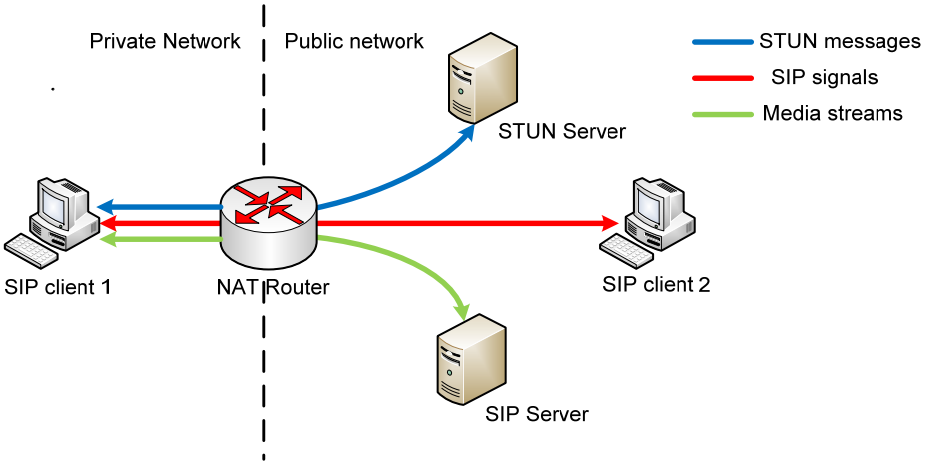


Figure 3-6. STUN mechanism

Depending upon the type of NAT, different address and port mapping schemes will have been used. STUN works primarily with three types of NAT: full cone NAT, restricted cone NAT, and port restricted cone NAT. An obvious drawback of STUN is that it does **not** work with a symmetric NAT, as this type of NAT will create a mapping based on internal IP address and port number **as well as** the destination IP address and port number. When the destination IP address of the SIP proxy is different from the address of the STUN server, then the NAT will create two different mappings using different ports for traffic to the SIP proxy and STUN server, thus *the mapping which STUN learned and which will be used during SIP call establishment messages is incorrect.*[19] As a result the SIP signaling will not be correct and the session will not be setup properly when a SIP client is behind a symmetric NAT.

STUN provides one solution for an SIP application to traverse the NAT, as it allocates a public IP address and port for the client and allows the client to receive packets from a peer with this transport address. However, an STUN server does not permit the client to communicate with all peers with the same transport address (public IP address and port).[20] This lead to the development of another solution that could address the problem, we describe this solution in the next subsection.

3.2.2 Traversal Using Relay NAT (TURN)

Traversal Using Relay NAT (TURN)[21], an extension to the STUN protocol, is designed to solve the symmetric NAT traversal problem. Because the problem caused by a symmetric NAT was that the external (public) IP address and port for the SIP client outside the NAT would be different if the packer were to be sent to another global IP address and port, the solution is for the TURN server to **relay** packets to and from other peers. In this way the mapping that the TURN client learns is correct and the packets that the SIP client sends will be relayed by this TURN server.

As shown in Figure 3-7, the TURN-enabled SIP client sends an exploratory request to the TURN server. A binding response containing the client's mapped IP address and port on the public side of NAT is sent back. This mapped IP address and port are used in both SIP call establishment messages and media streams.[19] The TURN server relays packets to the client when a peer sends data packets to the mapped address. Although a TURN server enables this client to communicate with other peers, it comes at a high cost to the provider of the TURN server as this server needs a high bandwidth connection to the Internet, since the amount of traffic across this connection is twice the volume of relay traffic - as all the traffic has to go both to the TURN server and from the TURN server to the relay target. Moreover, like STUN, TURN requires SIP clients to be upgraded to support its mechanism.

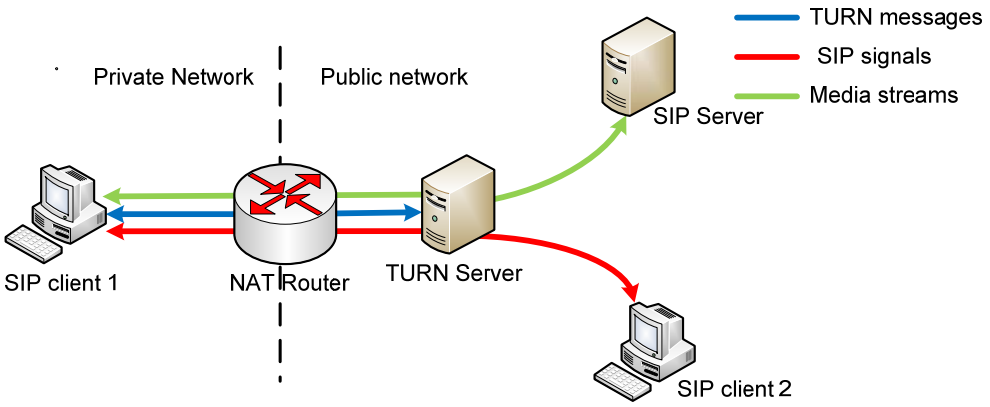


Figure 3-7. TURN mechanism

3.2.3 Interactive Connectivity Establishment (ICE)

ICE[22] is a form of peer-to-peer NAT traversal that works as an extension to SIP. ICE provides a unifying framework for using STUN and TURN around it. The detailed operation of ICE can be broken into six steps: (1) gathering, (2) prioritizing, (3) encoding, (4) offering and answering, (5) checking, and (6) completing.[22]

Gathering: Before making a call, the ICE client begins by gathering all possible local IP addresses and ports from interfaces on the host. These potential IP addresses and ports are called host candidates. Then, the client contacts the STUN server from each candidate to learn the pair of public IP address and port which are allocated by NAT for this candidate. These public IP address and ports are called server-reflexive candidates. Finally, the client contacts the TURN server and obtains relayed candidates.

Prioritizing: Once the client has gathered its server-reflexive candidates and relayed candidates, it assigns a priority value for each of them.

Encoding: After the gathering and prioritizing processes, the client constructs its SIP INVITIE request to set up the call. ICE adds host candidates, server-reflexive candidates and relayed candidates as candidate attributes in SDP attributes for the SIP Request message.

Offering and answering: The SIP network send the modified Request to the called terminal. The called terminal generates a provisional SIP response which contains the candidate information of the called terminal.

Checking: Through the above processes, the caller and called terminal have exchanged SDP messages. The caller and called terminal pair each of its candidates with a candidate from its peer. ICE uses a STUN transaction to check if a candidate pair works or not. This check is conducted in priority order and the highest-priority pair will be used for the subsequent traffic.

Completing: The caller generates the final check to its peer to confirm the highest-priority candidate pair as the one which will be used later. Finally, the media traffic begins to flow.

Although ICE combines the benefits of STUN and TURN without their disadvantages, it is still not a flawless solution and the drawback is both obvious and intolerable for the users. It inevitably increases call-setup delays -- as all of the gathering and checking takes place **before** the called terminal even receives the SIP INVITE. It also has disadvantages for the NAT, in that each of the candidates leads to the allocation of a server-reflexive candidate – thus taking up public IP address and port combinations that can not be used by another client inside the private network. While this might not be a problem for a single user at home, it can be a problem for a mobile operator who is using a NAT between their mobile packet data network and the public internet!

3.2.4 Application Layer Gateway (ALG)

An Application Layer Gateway (ALG) is an application specific translation agent which modifies the signaling to reflect the public IP address and port which are used by the SIP signaling and media streams (see Figure 3-8).

As ALG is an enhancement of a NAT/firewall, it is transparent for the users. The result is that no special additional mechanism or functions needed to be supported by SIP clients. However, in order to support ALG functionality, the NAT/Firewall needs a software upgrade or the user may even need to replace their existing NAT/Firewall with one that supports a SIP ALG.[19]

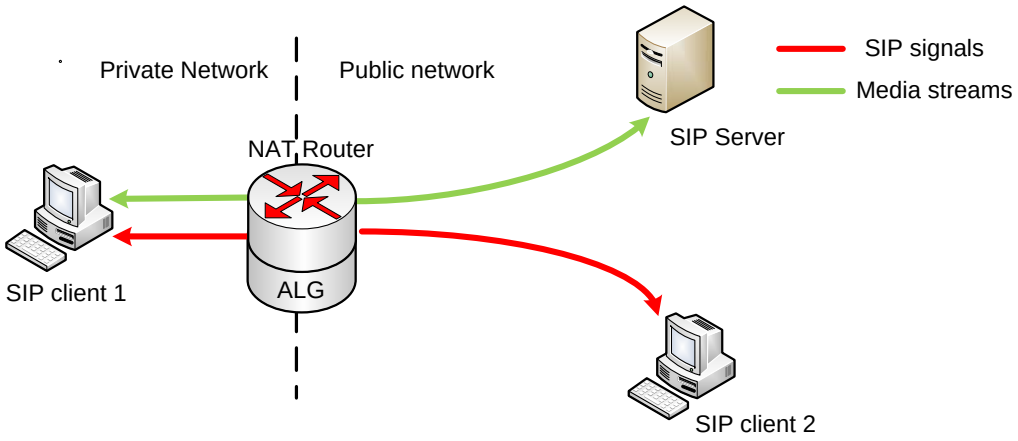


Figure 3-8. Application Layer Gateway mechanism

4 VPN and Security Protocols

This chapter provides a brief overview of virtual private network technology. IPSec and SSL/TLS, two of the main secure VPN protocols, are introduced and compared in Section 4.2 and 4.3. IPSec needs to have keys, thus either manually installed keys are used or some mechanism is needed to generate and exchange keys across the network. Internet Key Exchange is used to generate keys for the IPSec protocol suite and the Internet Security Association and Key Management Protocol is a key management protocol. How these two protocol work together with IPSec is presented in detail in Section 4.5.

4.1 VPN Overview

A **virtual private network** (VPN) is a private data network that makes use of the public telecommunication infrastructure, while maintaining privacy through the use of a tunneling protocol and security procedures.[23] VPNs enable corporations to securely access remote sites with "virtual" connections instead of private leased lines, thus their cost for connectivity can be as low as the cost of using the public internet infrastructure.

From a security stand point, VPNs guarantee security either by trusting the underlying delivery network or adding security schemes in the VPN itself.[24] Therefore, VPNs can be divided in two categories: trusted VPNs and secure VPNs.

In a trusted VPN, the customer uses no cryptographic tunneling. Such a VPN uses its own IP address and security policy. The VPN customer trusts the VPN provider and rents the leased virtual circuit to access the remote site.[23] Multi-Protocol Label Switching (MPLS) and Layer 2 Tunneling Protocol are frequently used to create a trusted VPN.

Secure VPNs use cryptographic tunneling protocols to encrypt traffic at the edge of one network and decrypt on the receiving side.[23] IPSec and SSL/TLS are the two main secure VPN protocols. We will introduce both of these protocols in next sections.

4.2 Internet Protocol Security (IPSec)

4.2.1 IPSec Architecture

Internet Protocol Security (IPSec)[25] is a suite of protocols for providing interoperable, high quality cryptographically-based security for IP communications. These protocols operate in the network layer to provide data source authentication, data integrity, confidentiality, and identity verification. The architecture of IPSec is shown in Figure 4-1.[26]

The Authentication Header protocol (AH) and Encapsulating Security Payload (ESP) are two different security protocols in IPSec. AH[27] is used to provide connectionless integrity and data original authentication for IP datagrams. ESP[28] is used to provide confidentiality, data origin authentication, and connectionless integrity. The main difference between AH and ESP is that the former provides only integrity protection, while the latter provides both encryption and integrity protection.

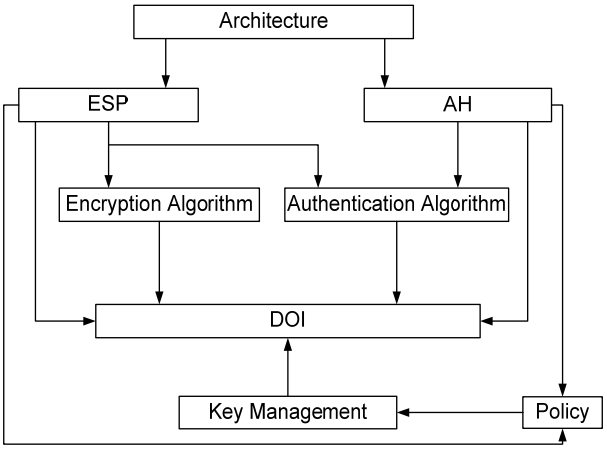


Figure 4-1. IPSec Architecture

4.2.1.1 Security Association

A security association (SA) is the set of shared security information between one device and another. It determines how the packets are processed. An SA includes cryptographic algorithms, keys, indicates if AH or ESP is to be used, key lifetimes, mode, and other security information which is used to encrypt and authenticate a one direction flow. Hence two SAs are needed to support a bi-directional flow.

4.2.1.2 Authentication Header

AH provides integrity for packet contents and the IP header. However, the protection provided to the IP header by AH is piecemeal as some mutable fields in the IP header which might be altered in transit **cannot** be protected by AH. These mutable fields include Service type, Fragmentation offset, TTL, and Header checksum.[27] The format of AH in Figure 4-2 and the fields are given Table 4-1. [27]

0-7 bit	8-15 bit	16-23 bit	24-31 bit
Next Header	Payload Length	Reserved	
Security Parameter Index (SPI)			
Sequence Number			
Authentication Data (variable)			

Figure 4-2. Authentication Header

Table 4-1. Field specification of AH header

Field Name	**Specification**
Next Header	Identifies the type of the next payload after the AH header
Payload Length	Specifies the length of AH header
Reserved	Reserved for future use
Security Parameters Index (SPI)	Enables the receiver to select the SA to which an incoming packet is bound
Sequence Number	Contains an increasing number
Authentication Data	Contains the integrity check value (ICV) for this packet

4.2.1.3 Encapsulation Security Payload (ESP)

The ESP header is inserted after the original IP header and before the transport layer protocol header in transport mode, or it can be inserted before an encapsulated new IP header in tunnel mode.[28] Transport mode and tunnel mode are described in Section 4.2.2. As noted earlier, ESP provides confidentiality, authentication, and integrity protection for a packet's payload. The format of and ESP packet is shown in Figure 4-3 and the fields are specified in Table 4-2.[28]

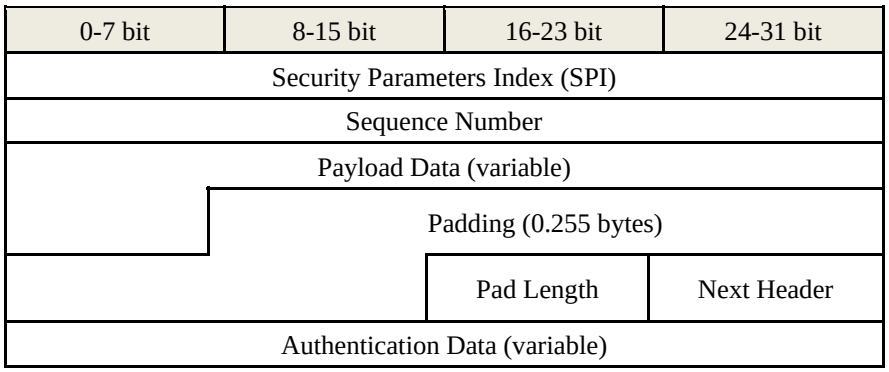


Figure 4-3. ESP packet layout

Table 4-2. Field specification of ESP packet

Field Name	**Specification**
Security Parameters Index (SPI)	Enables the receiver to select the SA to which an incoming packet is bound
Sequence Number	Contains an increasing number
Payload Data	Contains the data to be transferred
Padding	Pads the data to the full length of a block with block ciphers
Pad Length	Indicates the size of padding
Next Header	Identifies the type of data contained in the Payload Data field
Authentication Data	Contains the data used to authenticate the packet

4.2.2 IPSec Modes of Operation

IPSec has two operation modes, transport and tunnel, which provide security to transmitted data packets. Transport mode adds an IPSec header between the IP header and IP data to encrypt the data portion of each packet, while tunnel mode adds a new IP header and IPSec header before the original IP packet to encrypt the entire IP packet. Figure 4-4 and Figure 4-5 show the structure of an IP packet in transport and tunnel mode respectively. Transport mode is suited for end-to-end communication between two hosts and tunnel mode is suited for gateway-to gateway-communication.

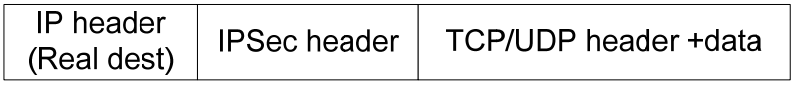


Figure 4-4. Transport Mode

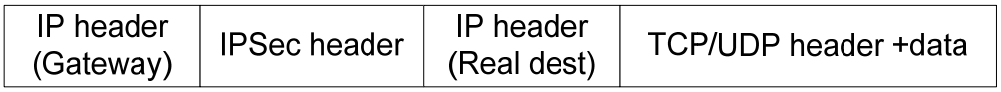


Figure 4-5. Tunnel Mode

4.2.3 Cryptographic Algorithms

ESP and AH are two separate mechanisms for protecting the data which is sent over an IPSec SA. In order to ensure compatibility of different implementations and that there is at least one algorithm that all implementations can support, a minimal set of algorithms are specified.[29] The authentication algorithms used in AH include: HMAC-SHA1, AES-XCBC-MAC, and HMAC-MD5. The authentication algorithms used in ESP include: HMAC-SHA1, AES-XCBC-MAC, HMAC-MD5, and NULL. The encryption algorithms used in ESP include: AES-CBC, 3DES-CBC, AES-CRT, DES-CBC, and NULL. The NULL algorithms do not provide an authentication or privacy – hence they are only included for testing purposes.

4.3 SSL/TLS

Transport Layer Security (TLS)[30] and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that provide privacy, authentication, and message integrity between two communicating applications. The difference between TLS and SSL is that TLS supports different encryption algorithms than SSL. In this thesis we will follow convention and refer to the protocol as SSL/TLS – even though we will only utilize TLS.

TLS works at the transport layer and is composed of two layers: the TLS Record Protocol and the TLS Handshake Protocol. The TLS Record Protocol is used for the encapsulation of various higher-level protocols. The TLS Handshake Protocol allows the server and client to mutually authenticate and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives data.

4.4 The comparison between IPSec and SSL/TLS

Both IPSec and SSL/TLS can provide authentication, data privacy, and data integrity. The significant and essential difference between them is that IPSec operates at the network layer, while SSL/TLS works on top of the transport layer as shown in Figure 4-6. Hence, IPSec secures **all data** flowing from one IP interface to another IP interface, which means nothing needs to be changed in the application layer to support IPSec. The application is not informed if IPSec is being used or not. Such a mechanism maximally alleviates the difficulty of application development and increases the flexibility of applications. In contrast, the application needs to be modified in order to support SSL/TLS. However, an advantage is that the application can know if it is using TLS or not. Additionally, the application can choose what keys, what algorithms, etc. that it uses.

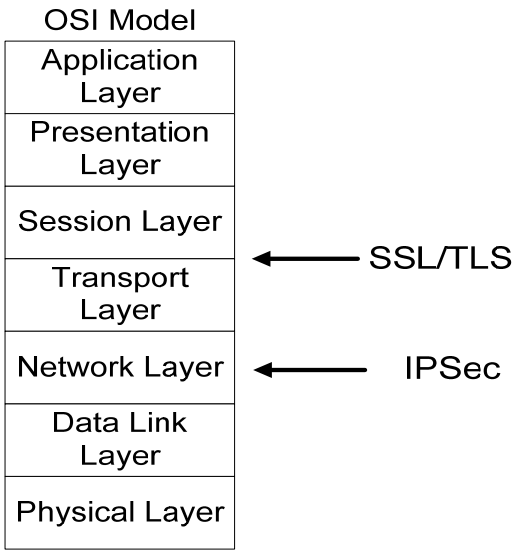


Figure 4-6. The location of IPSec and SSL/TLS

4.5 Key Management Protocols

Internet Key Exchange (IKE)[31] is a key exchange protocol used to generate keys for establishing a security association (SA) in the IPSec protocol suite. The parameters that are negotiated are documented in a separate document called the IPSec Domain of Interpretation (DOI).[26] This policy specifies some important parameters such as the type of algorithm, the key sizes and how the keys are derived, etc.

The Internet Security Association and Key Management Protocol (ISAKMP), is a key management protocol, hence it defines procedures and packet formats to establish, negotiate, modify, and delete Security Associations (SA).[32] IKE works with ISAKMP in IPSec. ISAKMP provides a common framework for key exchange and IKE provides mutual authentication and SA establishment.

IKE's operation can be split into two phases. Phase 1 establishes an authenticated, secure channel between two IKE peers. Phase 2 negotiates the IPSec SA and generates key material for IPSec[31].

5 VOIPsEC

In this chapter we propose VOIPSec as a NAT traversal solution for a VoIP application which builds up an IPSec VPN tunnel between the SIP participates and routes the traffic through NAT equipment. We describe how IPSec tunnels can be combined and how they will work with VoIP. As the IPSec mechanism was not designed to solve the NAT traversal problem, it is not completely compatible with NAT. In Section 5.2, we analyze these incompatibilities and explain how to over come them.

5.1 VPN and VoIP overview

One challenge of VoIP is how to route traffic through firewalls and NATs (as discussed in Section 3.1.3). Another problem is VoIP, as a computer-based technology, faces serious risks and attacks just as PCs have faced. Fortunately, a VPN tunnel can be used to solve both of these problems. After establishing a secure tunnel between the endpoints all the packets, both SIP signaling and RTP traffic, travel through the tunnel. As a result the traffic is protected *and* the peers are protected from traffic from other hosts (assuming that non-VPN traffic is rejected by the device's firewall).

VoIP media streams are very different from typical data traffic (such as file sharing, web browsing, or remote terminal access). Thus a voice conversation is broken up into small frames and encoded, then sent in RTP packets. These RTP packets are sent over an IP network from the source in order and with the same time interval as the frame sampling time – typically 20 ms. Each RTP packet has a unique sequence number and timestamp which are used at the receiver to place the RTP packets in the correct order and to detect losses. At the receiver RTP packets are *reassembled* and reordered based upon the timestamps and sequence numbers in order to maintain proper time consistency for the audio (media). As mentioned, VoIP traffic must be transmitted from the source to the destination within an acceptable (maximum) delay. Meeting this latency bound is more crucial to the perceived voice quality than other factors.

The VoIP traffic is sent in RTP packets which are encapsulated in UDP, thus an IPSec VPN is more suitable for this traffic than *an* SSL/TLS VPN; as IPSec can easily support UDP. While an SSL/TLS connection is initiated with TCP, which guarantees a stream of data sent from one host to another, without duplication or loss. However, TCP will require retransmissions in the event of packet loss (or damage), hence delaying all packets behind the lost packet, increasing delay, and increasing the variance of delays between RTP packets. Hence TCP is not suitable for real-time audio and video applications such as VoIP. As a result, a SSL/TLS VPN is not suitable for our application.

5.2 VOIPSec and NAT incompatibility

VoIP works with IPSec (VOIPSec) protects the RTP packets from end-to-end or gateway-to-gateway.[33] Using an IPSec VPN reduces the threat of a man-in-the-middle attack, vulnerability to packet sniffers, and the impact of voice traffic eavesdroping as it encrypts data before it traverses the public network. An IPSec VPN can use existing network connections to access corporate networks, which saves the cost of building point-to-point links (or renting MPLS or leased lines). However, IPSec was not originally designed to traverse NATs. In other words, IPSec and NATs are not completely compatible. The two main incompatibilities are discussed below.

5.2.1 Operation Mode

IPSec has two modes of operation: AH and ESP. However, IPSec AH is not compatible with NATs because AH protects the packet contents and IP header while NATs exchange the (internal) private IP address with an (external) public IP address, thus modifying the header. Figure 5-1 illustrates an AH header in both transport mode and tunnel mode. In transport mode, AH provides integrity for the payload and IP header and inserts a new AH header between the original IP header and the payload. In tunnel mode, AH encapsulates the entire IP packet and inserts a new IP header. On the other side, the recipient receives the packets, then authenticates the sender by recalculating the hash. The packets will be discarded if the hashes do not matched, which would indicate that the packets have been tampered with. However, because the NAT modifies the IP header, replacing the private IP address with a public IP address, the recipient will discards all of the packets as the calculated hash will not match the expected hash.[34]

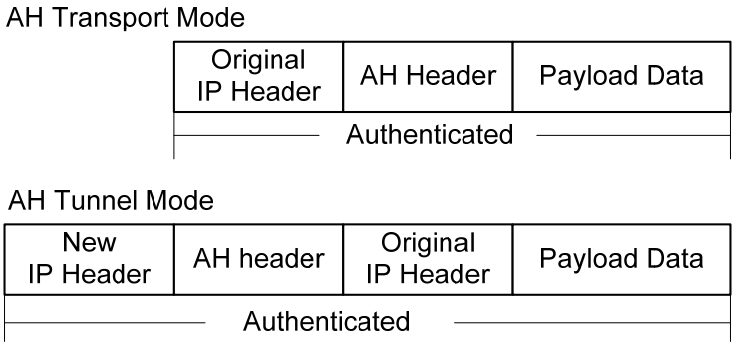


Figure 5-1. Authentication Header

Figure 5-2 illustrates the ESP mechanism in different connection modes. In transport mode, ESP only encrypts the payload and provides connectionless integrity for the packet's contents, but **not** for the IP header. In tunnel mode, the ESP encrypts the IP packets and inserts a new IP header. The new IP header will be modified when the packet traverses the NAT, but the original IP header, which was encrypted by ESP, will not be altered. The recipient decrypts the packets and forwards the original IP packets, which is contained in the original IP header. Therefore, the ESP mechanism is more suitable for the NAT system.[34]

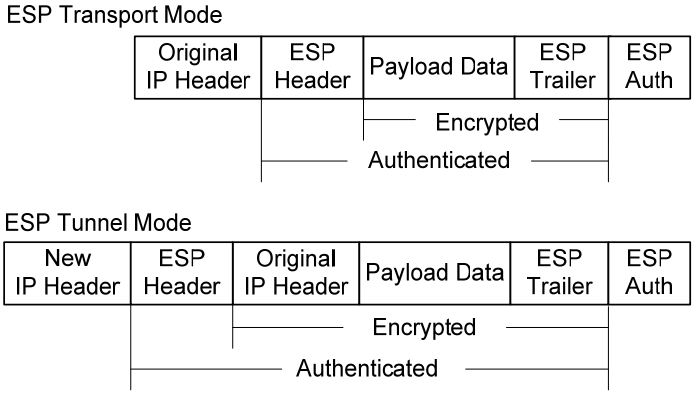


Figure 5-2. Encapsulating Security Payload

5.2.2 IPSec and NAT

Another issue about IPSec and NAT is that most NAT devices bind several (internal) private IP addresses with one (external) public IP address. This binding is based upon both IP address and port number. However, the IPSec encapsulated packet encrypts the payload of transport layer protocol. As a result the NAT device cannot access the transport layer header, i.e., it can not learn the transport layer port number! Nor can the NAT change the IP address and port number after IPSec ESP processing. If there is only one endpoint, for example, the VoIP device, behind the NAT, then the NAT could simply replace the private IP address with a public IP address and there is no need to modify the port. However, if there is more than one endpoint behind one NAT trying to communicate with the same server, for example, multiple VoIP devices negotiating with the same SIP proxy, an NAT mapping problem is inevitable since the NAT device has to create a unique mapping with a public IP address and port number for each endpoint. Fortunately, UDP encapsulation of the ESP packet solves this problem by wrapping the IPSec ESP packet with a duplicate UDP header as illustrated in Figure 5-3. The NAT device modifies the new **unencrypted** IP and UDP headers of the UDP-encapsulated ESP packet without changing the ESP authentication and encryption. The UDP-encapsulated packet is sent over UDP port 4500.[35]

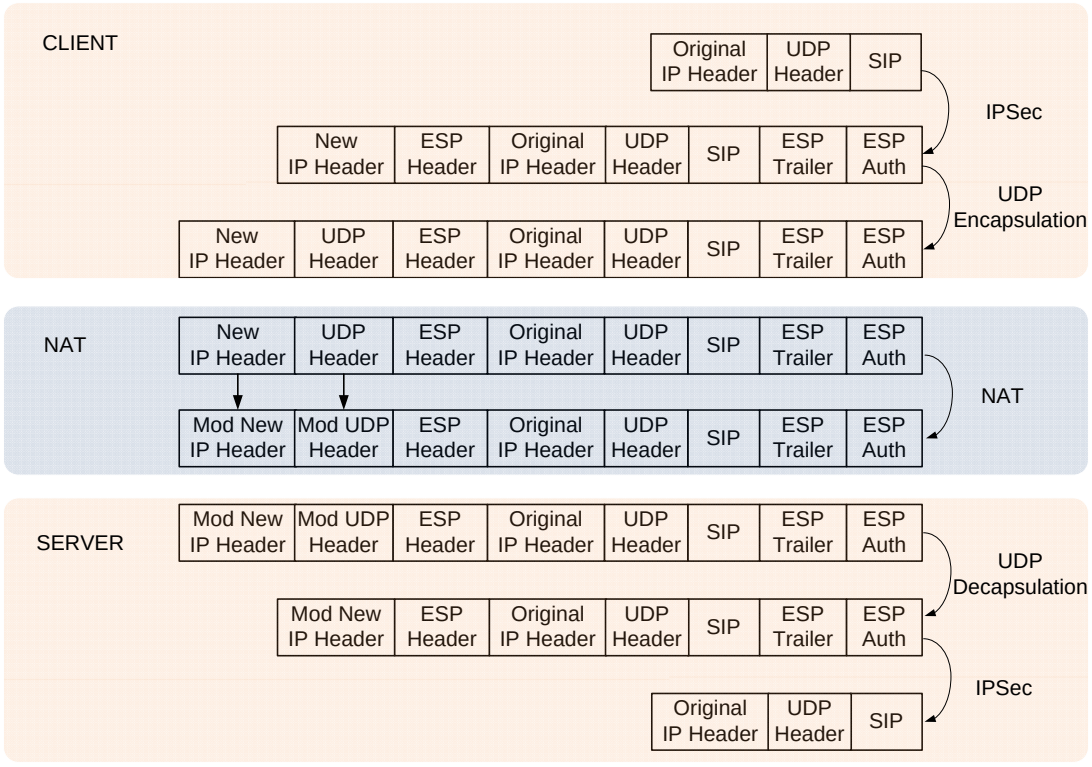


Figure 5-3. UDP Encapsulation ESP packet in Tunnel mode

6 SRTP and MIKEY

SRTP is an extension of the RTP profile, to provide a security for RTP streams. In this chapter we describe SRTP and show how our proposed extended VOIPSec solution can work with SRTP. The details of SRTP are given in section 6.1. Following this, section 6.2 introduces MIKEY, a key management protocol used to generate the keying materials needed by SRTP.

6.1 SRTP

The Secure Real-time Transport Protocol **(**SRTP) is an extension of the RTP profile. SRTP provides the framework for encryption, message authentication and integrity, and replay protection of RTP and RTCP streams.[36] SRTP is independent of the network and transport layer. It protects the traffic on the application layer. SRTP intercepts RTP packets and forwards SRTP packets to the transport layer on the sending side, and on the receiving side SRTP intercepts the SRTP packets and forwards RTP packets. The format of an SRTP packet is shown in Figure 6-1.

Figure 6-1. SRTP packet architecture

The SRTP packet consists of: fixed RTP headers, and encrypted RTP payload, (optional) MKI, and Authentication Tag. The MKI (Master Key Identifier) identifies which master key was used to derive the session key that should be used with this packet. The Authentication Tag contains message authentication data and provides authentication of the RTP header and payload. It protects against an attacker sending modified packets or inserting additional packets.

The Advanced Encryption Standard Counter Mode (AES-CM) encryption method is mandatory to implement for SRTP. AES in f8-mode (AES-f8) is an optional encryption method and is used by Universal Mobile Telecommunications System (UMTS) 3G mobile networks. HMAC-SHA1 as defined in RFC 2104 [37] is the pre-defined authentication algorithm for SRTP. Additional encryption algorithms and authentication algorithms can be used if both peers support them and wish to use them. The details of selecting suitable algorithms for either use lies outside the scope of this thesis.

6.2 MIKEY

Multimedia Internet KEYing (MIKEY) is a key management protocol and was developed especially for real-time-applications running over SRTP. MIKEY supports three different key agreement mechanisms: pre-share key, public key, and Diffie-Hellman key exchange.[38]

6.2.1 Pre-shared Key

In the pre-shared key method, a pre-shared secret key is used to derive session keys for encryption and integrity protection. The pre-shared key method is the most efficient way to handle key transport due to the fact that only a small amount of data has to be exchanged. However, it is not easy to share secret keys with a large group of peers, leading to scalability problems.[38] However, this mechanism may be very suitable for a small company or group of friends.

6.2.2 Public-keys

Unlike the pre-shared key method, in the public-key method every use has a pair of keys: a public key and a private key. The sender encrypts the message with the recipient's public key, which is published to everyone. Only the recipient knows its own private key. Hence only the recipient can decrypt the message. Public-key cryptography can solve the scalability problems, but it is more resource-consuming than the pre-shared key approach.[38] It also assumes that there is some way that a sender can find out the intended recipient's key and know that this key is actually the currently valid key to use for this recipient.

6.2.3 Diffie-Hellman Key Exchange

Diffie-Hellman (DH) key exchange is a way that two parties can agree upon a common secret. In this approach both of the parties contribute to the secret and no other party can learn this secret – even if they eaves drop on the communication between the parties. This method provides perfect forward secrecy. However, its resource consumption is even higher than the public-key method.[38] Additionally, this method is vulnerable to man-in-the-middle attacks.

7 Objective

This chapter briefly presents the problem to be solved in this thesis project. The mechanism underlying the VOIPSec solution and how to measure and evaluate the proposed solution's performance are introduced.

7.1 Implementation : Enable NAT traversal as well as make a secure VoIP call

The Internet has recently been experiencing massive growth in real-time multimedia applications, such as video and audio streaming. VoIP technology, an emerging trend in telecommunications, has become attractive due to its ability to increase scalability and availability while reducing costs. However, the problems and risks that accompany VoIP technology cannot be neglected. This thesis project deals with two main challenges of VoIP: NAT traversal and security. In this project, we propose a complete and feasible solution for NAT traversal and VoIP security. This solution is applied to VoIP communication between two independent VoIP terminals.

The main idea underlying the VOIPSec solution is to establish IPSec VPN tunnels between a SIP server & a SIP client and between SIP clients. IPSec VPNs are used for both SIP signaling and real-time media traffic. The clients are assumed to potentially be behind more than one NAT. VPN tunnels are used to construct a logical communication network for communication between SIP clients and SIP servers. These clients and servers can be attached to different networks. SIP clients (e.g., SIP terminals) send signaling and data traffic on this logic network to avoid VoIP NAT traversal difficulties.

Once the IPSec VPN tunnel has been established, both SIP signaling and RTP media packets are protected by sending packets only over this VPN. However, IPSec is a network layer security protocol, which means it can protect traffic between the routers or host that are the VPN tunnel endpoints. As a result both the SIP and RTP packets would be available to any application running on the end node if only IPSec is applied. In order to improve the data protection offered, SRTP is utilized to protect the RTP content end-to-end. This is possible because SRTP is a transport layer mechanism implemented at the application layer and its destination is an individual application.

The proposed VOIPsec solution affects the whole communication process, which includes three phases: Call Establishment, Conversation, and Call Termination. The call establishment and call termination phases involve SIP signaling stream to set up or terminate calls; while the conversation phase uses RTP for media transmission. As noted above we will use SRTP to protect the RTP traffic application to application. To add additional security to the SIP signaling we can use TLS to protect this traffic application to application. Note that TLS is suitable for the signaling as we want it to be sent in order and reliably.

7.2 Measurement

The performance of the VOIPSec solution was evaluated by measuring the delays in the VPN setup process, the SIP call establishment and termination processes, and the end-to-end delay of the

RTP packets. It is important to note that VOIPSec can add additional delay to the VPN establishment phase, Call setup phase, Call termination phase, or media processing. The most important delays are the per RTP packet delays during a call – as this will determine whether the solution will be acceptable or not; while additional delays at VPN establishment and the call setup phase could be annoying to the user.

The main results of this research have been two proposals for VOIPSec solutions (at the network level and application level respectively). The test bed and performance measurements of these two solutions are described in the following chapter.

8 Implementation and Measurements

In this chapter, measurements and evaluations of the VOIPSec and extended VOIPSec solutions will be presented. These measurements focus on quantifying the delay of both the trust establishment and the dialogue processes. The proposed VOIPSec solutions are evaluated using the test bed described in section 8.1. The evaluations are divided in different cases, where the clients in the test are in different locations. The cases were chosen to reveal the performance in different situations and to determine the factors that influence the delay of our solutions. A detailed analysis and discussion of the measurement results will be presented after the measurements themselves have been presented.

8.1 Test bed

Figure 8-1 shows the test bed used in this project. Details of the computers and routers that were used to construct this test bed are given in Table 8-1. It contains four interconnected networks that include subnets 192.168.1.0/24 and 192.168.3.0/24, which are two separate domains, as well as a subnet 192.168.2.0/24, which is a subdomain of 192.168.1.0/24. Thus hosts on the subnet 192.168.2.0/24 must go through two NAT routers to reach the Internet. There are several client terminals which act as both SIP and VPN clients in the different networks. The VPN server (Openswan 2.6.20) and the SIP server (SIP Express Router) are installed and run on a single independent machine which has one public IP address. SIP clients use an open source Linux SIP user agent Minisip; while VPN clients use Openswan, an implementation of IPsec for Linux.

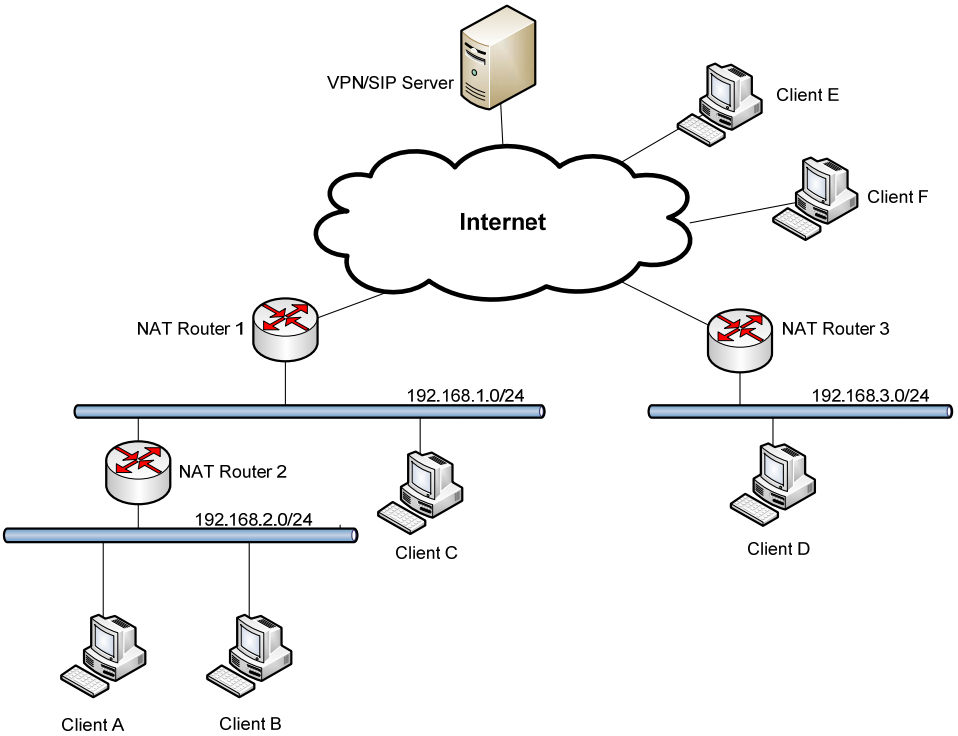


Figure 8-1. Test Bed

Table 8-1. Test bed elements

Item	Description
Client A	Dell™ Dimension™ 8400; Intel® Pentium® 4 processor 2.80GHz; 1GB RAM, Broadcom NetXtreme 57xx Gigabit Controller
Client B	Dell™ PowerEdge™ SC420; Intel Celeron® processor 2.53GHz; 504 RAM; Broadcom NetXtreme 5751 Gigabit Controller
Client C	Dell™ Dimension™ 8400; Intel® Pentium® 4 processor 2.80GHz; 1GB RAM, Broadcom NetXtreme 57xx Gigabit Controller
Client D	IBM Think Pad T61; Intel® Core Duo processor T7300 2GHz; 1GB RAM, Intel 82566MM Gigabit Network Connection
Client E	IBM Think Pad T61; Intel® Core Duo processor T7300 2GHz; 1GB RAM, Intel 82566MM Gigabit Network Connection
Client F	IBM Think Pad T43; Intel® Pentium® M(Dothan) 750 1.86G; 512M RAM
VPN/SIP server	Dell™ PowerEdge™ SC430; Intel® Pentium® D processor 2.80 GHz; 2G RAM
NAT Router 1	Linksys® RV082 10/100 8-Port VPN Router - This router was connected via a 10 base T link to the ISP: STING Network.
NAT Router 2	NetGear® Super-G Wireless Router WGT624 v2 - This router was connected via a 100 base T link to subnet 192.168.1.0/24 and 192.168.2.0/24
NAT Router 3	TP-LINK® TL-WR641G+ - This router was connected via a 100 base T link to the ISP: TELE2.

8.2 VOIPSec-Network level solution

8.2.1 Performance of NAT Traversal

The first step of VOIPSec is VPN setup. This enables the VoIP participants to pass traffic through the NATs and communicate securely with each other. Because the time taken for VPN setup is nicely separated from the rest of the call processing and because it additively influences the SIP call setup process, we need to measure its performance. Section 8.2.1.1 presents the detail of the measurements and evaluations, while section 8.2.1.2 discusses the results of measurements and analyzes the elements that affect the performance.

8.2.1.1 Measuring Performance of NAT Traversal

An IPSec VPN tunnel is used to route the traffic passing through the NATs.

Figure 8-2 is a sequence diagram indicating the IKE messages that establish the IPSec tunnel between the VPN client and server, as well as the corresponding delays. As pre-shared key authentication is the simplest mechanism and has less delay than the public key mechanism, we use pre-shared key authentication with Main Mode for the first phase of IKE to establish IKE SA. Quick mode in IKE phase 2 establishes one or more IPSec SAs for the target protocol ESP between the nodes using the keys derived in phase 1. The messages exchanged in phase 2 are protect by the IKE SA which is created from IKE phase 1[39].

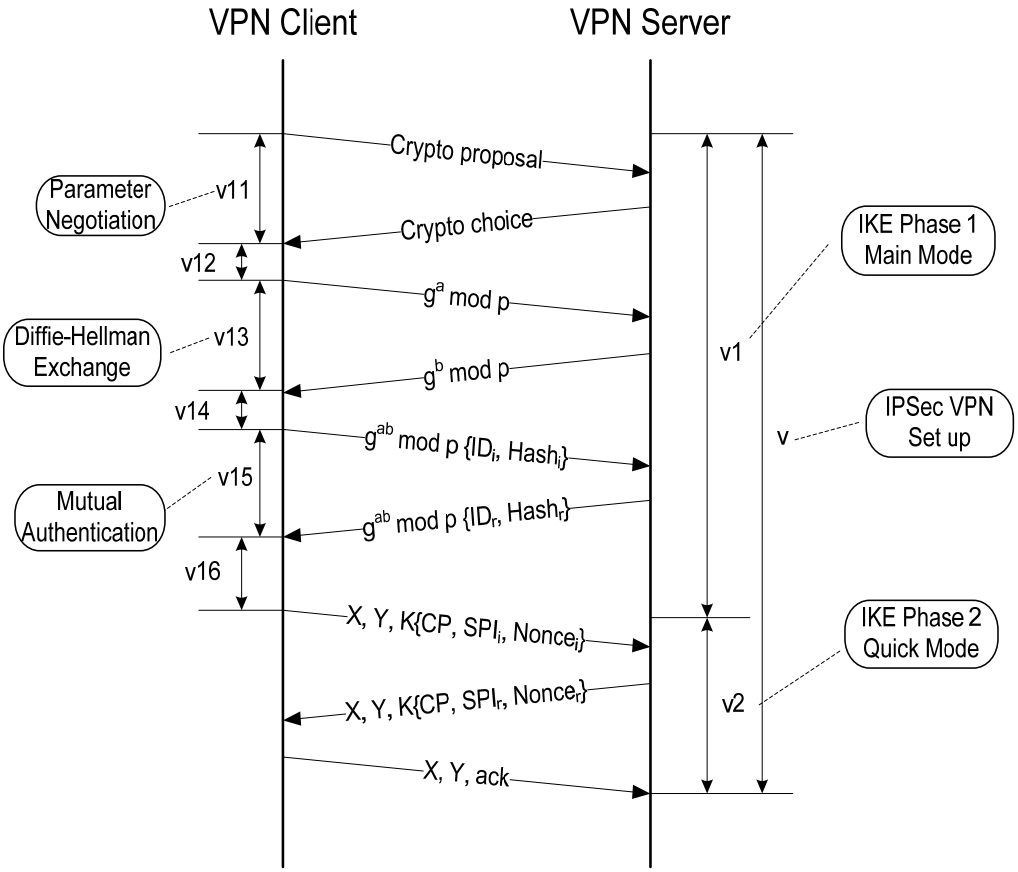


Figure 8-2. IKE Messages Flow [39]

The payloads contained in the message are as follows[39]:

ID_i	Identification - Initiator
ID_r	Identification - Responder
$Hash_i$/$Hash_r$	Hash payload of key, Diffie-Hellman values, nonces, crypto choices, cookies
X	A pair of cookies form phase 1
Y	32 bit message ID
K	Integrity key of authentication and Encryption key

The intervals v11 and v13 represent the time required for the exchange of the first two request/response pairs of messages, IKE SA initiation exchanges, which negotiate cryptographic algorithms, exchange nonces, and do a Diffie-Hellman exchange. The third pair of messages, during the time interval v15, authenticates the previous messages, exchange identities and certificates, and establishes the first IKE SA. Quick mode is essentially an SA negotiation and an exchange of nonces that provides relay protection.[31] The delay of IKE phase 1 (v1) contains the parameter negotiation period (v11), the Diffie-Hellman exchange period (v13), and the mutual authentication period (v15). The time v2 is the IKE phase 2. The total delay of the IPSec tunnel setup is represented by v. Note that v12, v14, and v16 represent the ISAKMP processing delay.

The VPN tunnel NAT traversal, a requirement of the VOIPSec solution, has to be accomplished **before** the VoIP call setup. Adding the IPSec VPN establishment process before the actual VoIP call adds delay. In this thesis we will refer to this as the **IPSec VPN Set up Delay**. Note that this tunnel could in practice be sent up when the device wants to register with its SIP registrar.

Intuitively the number of NATs between the VPN client and VPN server will influence the time required to establish a VPN tunnel. Hence we divide our VOIPSec NAT traversal measurements into two tests. In test 1, the VPN client is behind a single NAT. Table 8-2summarizes the IPSec setup delays for five test runs and Figure 8-3 plots the delay for each test run. In test 2, there are two NATs between the VPN client and VPN server. Table 8-3 summarizes the IPSec setup delays for five test runs and Figure 8-4 plots the delay for each test run. While five test runs are **not** sufficient to compute results with a 95% confidence, the small standard deviation from the mean indicates that the measurement for each run was close to the mean. For the rest of our discussion we will assume that the measurements are normally distributed and that the mean is representative of the delay.

Table 8-2. IPSec VPN Setup Delay measurements for test 1 (single NAT)

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Mean**	**Standard Deviation (σ)**
v11(ms)	3.5	3.4	3.4	3.3	3.8	3.5	0.2
v12(ms)	5.0	6.7	5.0	3.7	4.9	5.1	1.1
v13(ms)	17.7	16.6	16.5	17.7	16.7	17.0	0.6
v14(ms)	4.4	6.0	4.5	3.7	4.4	4.6	0.8
v15(ms)	3.1	3.0	3.0	3.1	4.0	3.2	0.4
v16(ms)	5.1	6.7	5.4	4.2	5.0	5.3	0.9
v1(ms)	38.7	42.4	37.8	35.7	38.8	38.7	2.4
v2(ms)	31.9	31.7	30.9	28.2	29.8	30.5	1.5
v(ms)	70.7	74.2	68.7	63.9	68.6	69.2	3.7

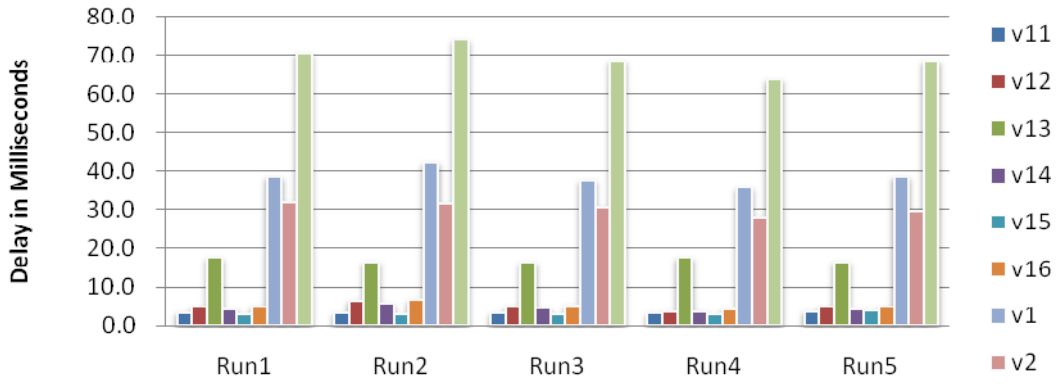


Figure 8-3. IPSec VPN Setup Delay measurements for test 1 (single NAT)

Table 8-3. IPSec VPN Setup Delay measurements for test 2 (two NATs)

	Run 1	Run 2	Run3	Run4	Run5	Mean	σ
v11 (ms)	6.0	6.7	7.0	5.7	6.3	6.3	0.5
v12 (ms)	7.4	4.1	4.7	5.1	6.0	5.5	1.3
v13 (ms)	18.9	19.6	20.8	21.9	19.1	20.1	1.3
v14 (ms)	4.4	4.9	4.6	5.3	5.0	4.8	0.4
v15 (ms)	4.9	6.0	7.0	5.2	5.6	5.7	0.8
v16 (ms)	5.0	4.8	4.9	5.1	6.0	5.2	0.5
v1 (ms)	46.6	46.0	49.0	48.3	48.0	47.6	1.2
v2 (ms)	30.8	32.2	31.6	33.2	31.9	34.9	0.9
v (ms)	77.4	78.3	80.6	81.5	79.9	79.5	1.7

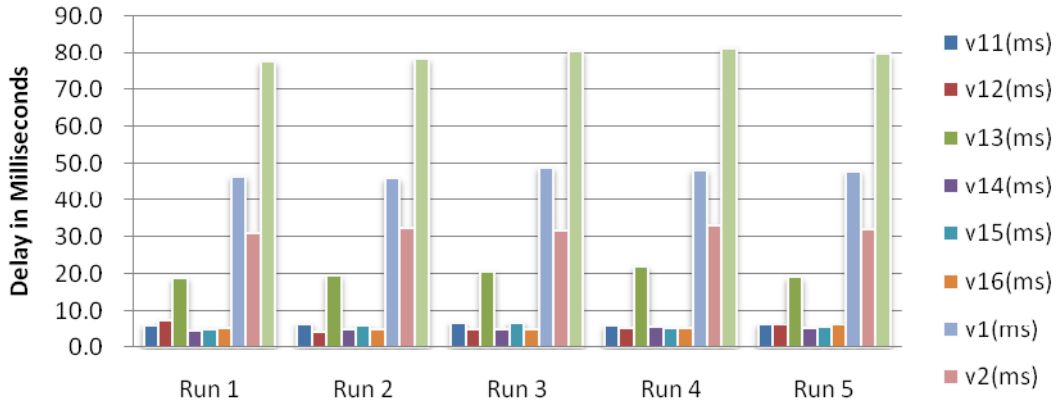


Figure 8-4. IPSec VPN Setup Delay measurements for test 2 (two NATs)

8.2.1.2 Performance analysis

The delay specified as v11 is the time taken for the first IKE exchange pair. From Table 8-2 and Table 8-3, the average delay of test 1 is 3.5 ms, while the average delay of test 2 is 6.3 ms. This v11 delay is due to the time it takes to exchange the algorithms, hashes, and other parameters to secure the IKE communications between the initiator and recipient. The initiator states the cryptographic algorithms it supports for the IKE SA. The responder chooses a cryptographic suite from the initiator's offering and sends its choice back to initiator.

The delay specified as v13 is the time taken for the second IKE exchange pair. This delay in test 1 (summarized in Table 8-2) ranges from 16.5 ms to 17.7 ms with a mean of 17.0 ms. In test 2 this delay has a slightly higher value, ranging from 18.9 ms to 21.9 ms, with a mean of 20.1 ms. This delay is mainly due to the Diffie-Hellman exchange, which is used to generate a shared secret keying material, and the passing of the nonces. The time to perform the Diffie-Hellman exchange will vary depending upon the computational capabilities of the hosts, while the passing of nonces will largely be determined by the network latency assuming that the nonce can be computed in advance. If the nonce can not be computed in advance, then the time required to generate a nonce can be dependent upon the performance of a local source of a random number.

The delay specified as v15 is the time taken for the third IKE exchange pair. This average delay is around 3.2 ms in test 1 and 5.7 ms in test 2. During this period of time, the initiator and the responder send their own identity and authenticate the other's identity.

The delay specified as v2 is the period of time taken to negotiate the IPSec SAs to set up the IPSec tunnel. All messages in this period are cryptographically protected using the session key negotiated in IKE phase 1. The nonces exchanged in this period prevent replay attacks from the generation of fake SAs.

From the performance shown in Table 8-2 and Table 8-3, we can observe that in the delays related to the negotiation between the initiator and responder, v11, v13, v15, v1, and v2 -- there is a relatively large difference between test 1 and test 2. The delays of test 2 are bigger than those of test 1 due to the network infrastructure differences in these two tests. The VPN client in test 1 is behind one NAT compared to the client in test 2, which behind two NATs. The IP address of the client will be modified and replaced by a new mapping IP address when it passes through the NAT. In test 1, the packet from client only changed its IP address once because there was only one NAT. In test 2, the IP address was modified twice, as the packet traversed two NATs. The round trip delay to send a UDP packet which contains 300 bytes payload from the source to the destination in the two cases were 4.2 ms and 6.8 ms. As the processing at the end nodes is the same in both test, we can assume that the difference in the delays is solely due to the difference in the NAT traversal process having to be done two as often in test 2 as in test 1. Hence the additional NAT traversal delay is the primary reason for the difference in delay between the two tests. From this we can estimate that the round trip NAT processing delay was 2.6 ms. However, despite going through two NAT vs. one NAT the (mean) total delay is 79.5 ms versus 69.2 ms. Although this delay is additive to the total delay to initiate a call; in practice this delay should not be perceived by the end user.

8.2.2 Performance during the Dialogue

In this thesis, we group the call initiation, RTP media exchange, and call termination into the VoIP dialogue. Today, quality of service (QoS) is a major issue in VoIP implementations. High QoS is mainly a demand for corporate LANs, private networks, and intranets – fortunately these users are generally communicating via an intranet, rather than the internet; hence the basic packet loss rate is low and the end-to-end delay can generally be controlled. In both corporate and private settings VoIP quality generally has two main aspects: the time required for call setup and the user perceived voice quality during a call.

The call setup starts when the caller picks up the phone. The caller waits to get a dial tone, dials the callee's number, waits for the network to connect the call to the callee's phone (when the ringing starts), and waits for the callee to pick up their phone. In addition, for a VoIP call the caller must also wait for their SIP user agent and the callee's user agent to negotiating the call parameters before media transmission can begin. Call setup performance is evaluated by measuring the time to perform a call setup. Here we will assume that the callee automatically answers the call, so there is no waiting for them to answer (and no variance in the waiting time for this operation).

Good voice quality means both the VoIP participants are able to speak and listen to a clear and continuous voice signal without unwanted noise. This quality depends on the following factors: latency, jitter, and packet loss.

Latency (packet delivery delay) is the time between the moment an RTP packet is transmitted and the moment it arrives at the destination. Large delays can lead to acoustic echoes (in the case of a hands free conference phone). However, when using an audio headset there is no acoustic echo. Hence the main effect of latency is the perception of having to wait for the other party to speak or in

the worst case (with high latency) the conversation degrades to simplex communication. Latency is generally measured in milliseconds (ms). In this experiment, we focus on one direction latency measurements of the time taken for a packet traveling from the source to the destination. Thus our measurements of latency explicitly exclude the time required to collect a speech frame, the time to encode this frame, the time to turn this into an RTP packert, the time to receive the RTP packet (once it has arrived at the destination host), the time to decode the speech frame, and the time delay in the playout (dejitter) buffer and in the audio device before the audio is presented to the listener.

Jitter is the statistical variation of RTP packet inter-arrival times. Jitter is only a problem when the amount of jitter exceeds the receiver's dejitter buffer.

Packet loss can occur for a variety of reasons including link failure, too much traffic on the network causing routers to drop packets, Ethernet problems, the occasional misrouted or damaged packet, and packets arriving at the destination beyond the latest time that they are needed (i.e., the packet arrives too late to play).

Section 8.2.2.1 presents the details of the measurements and evaluations for both call setup time and voice quality. Section 8.2.2.2 discusses the results of measurements and analyzes the factors that affect the performance.

8.2.2.1 Measuring Performance during the Dialogue

After the IPSec VPN tunnel establishment between the SIP client and SIP server, all the packets are encrypted and encapsulated by the endpoint at one side of the tunnel. The contents of the packets in the tunnel are opaque and cannot be modified without detection. The encapsulated packets are decapsulated and decrypted by the end point when they arrive at the other end of the tunnel. The encryption and decryption process for each packet will increase the delay when exchanging messages. In this section we describe our measurements of this additional delay.

Because the dialogue process is set up within a VPN tunnel, all messages to initiate a session or terminate a session are encrypted by an encryption key which was created during the IPSec tunnel setup process. The message flows during the dialogue between the caller, server, and callee are illustrated in Figure 8-5. When a caller wishes to initiate an SIP session with the callee; they do this by sending an INVITE request, which asks the SIP server to forward this INVITE to the caller in order to set up a session. The INVITE request is encrypted and encapsulated by ESP leading to an encryption delay (a11) before the caller can send the request. The INVITE request is received and forwarded by the SIP server. The process between receiving and forwarding contains two phases: decrypting the ESP packet from the caller and encrypting the packet with the encryption key used for the VPN tunnel between the SIP server and the callee. The SIP server is assumed to immediately forward the SIP INVITE. Eventually, the ESP packet arrives and is decrypted at the receiving side, which can choose to accept the invitation or reject it. SIP processing includes time for the callee to decrypt the ESP packet and process the incoming SIP messages. The 100 Trying response is sent after the request has been received by the server. When the callee's phone rings and alerts the callee of the INVITE request, a 180 Ringing response is sent from the callee and forwarded by the SIP server to the caller. After a small amount of SIP processing, a local ringtone is generate at the caller side[*]. The

[*] Note that we do not consider the case of early media where the caller can actually begin to receive associated with the callee before the session is established – for example, to hear the calleee's phone rining.

200 OK response is sent when the callee answers, this creates a dialogue between the caller that issued the INVITE and callee. The caller responses with an acknowledgement (ACK). The time period between when the caller receives the 200 OK message and the callee receives an ACK is called the Caller Transmit time (a2). The time period from when the callee sends the 200 OK and receives an ACK is called the Callee Transmit time (c2).

Note: In our measurements we have assumed that the SIP server acts as an incoming SIP proxy for the callee, allowing the callee to change their point of network attachment or to change the device(s) that they wish to use to receive SIP INVITEs. The SIP proxy could process the received request using a Call Processing Language script or other means to determine if an INVITE should be delivered and where this INVITE should be delivered to. For the purposes of this thesis we assume that the callee has registered only a single SIP client with the SIP registrar (which is co-located with the SIP proxy) and that the SIP proxy has cached the IP address that this client has registered, thus there is no delay due to the proxy needing to decide where to forward the INVITE.

In our proposed VOIPSec solution, media traffic which carried by RTP packets is relay by an RTP proxy. We use open source software called MediaProxy as a media relay for the RTP streams. The time gap called Packetization Delay and Encrypting Delay (c31) before sending each RTP packet is the time required to packetize the media traffic and encrypts it using ESP. The **uplink** indicates the RTP stream(s) from caller to callee. The **downlink** indicates the inverse direction of the uplink, i.e., it is the media stream(s) from callee to caller. Uplink RTP Delay (a4) and Downlink RTP Delay (c3) are the average delay of RTP packets in the uplink direction and downlink directions respectively.

The SIP BYE request is used to terminate a session between caller and callee. The Termination Delay (c5) is the time required to terminate and quit the SIP session.

Using the test bed (shown in Figure 8-1), the test can be divided into six different cases, which are described in Table 8-4.

Table 8-4. Test cases for measurement of the dialogue performance

Case		Description
1	Client A → Client B	Both the caller A and the callee B are behind two NATs.
2	Client A → Client D	Caller A is behind two NATs, while callee D is behind one NAT (NAT router 3).
3	Client A → Client E	Client A, behind two NATs, makes a call to Client E. Client E is connected to the public network, but communicates with the VPN/SIP server through IPSec tunnels.
4	Client C → Client D	Both caller C and callee D are behing one NAT.
5	Client C → Client E	Caller C, behind one NAT, calls callee E, which has a public IP address.
6	Client E → Client F	Both caller E and callee F are connected to the public network

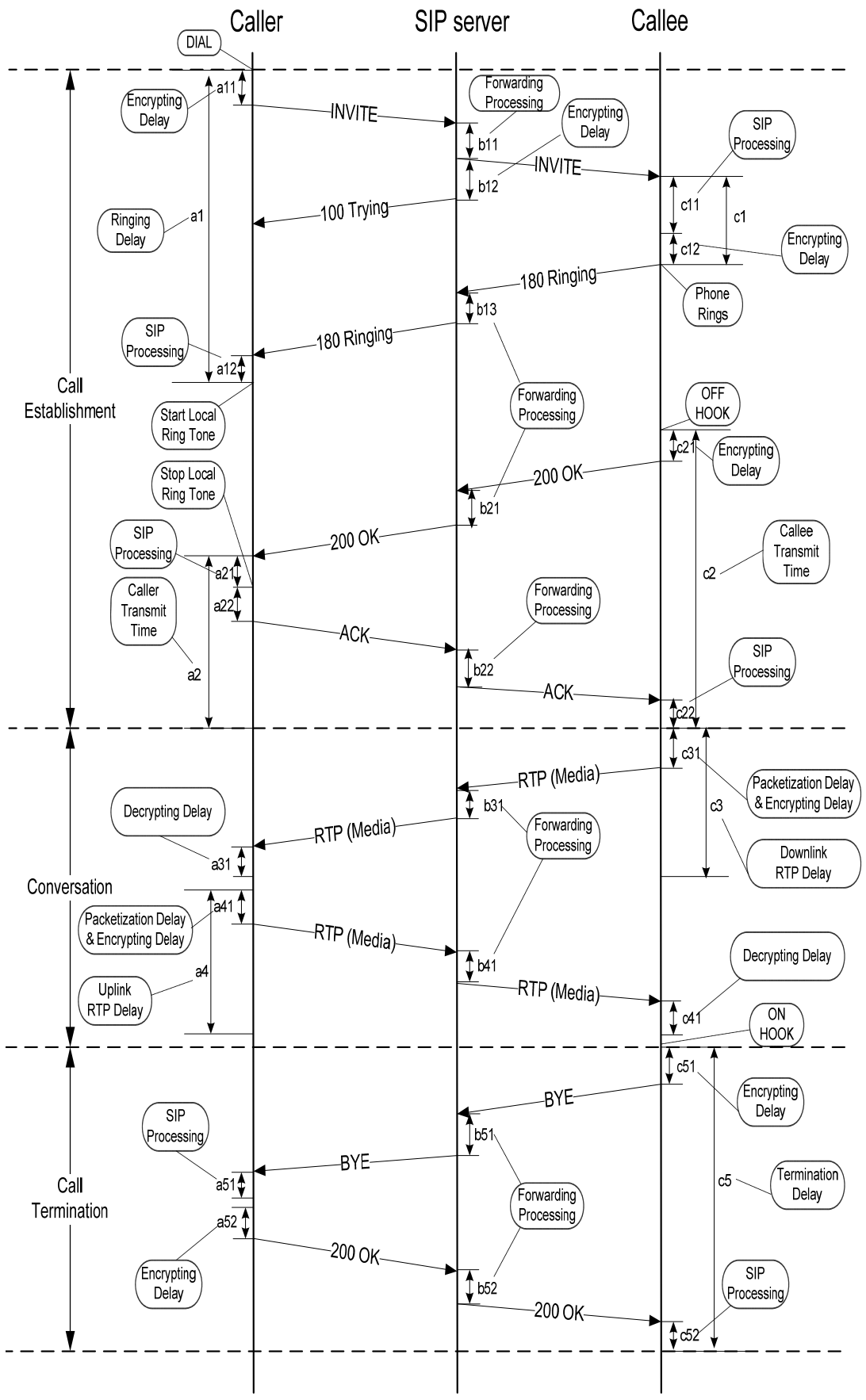


Figure 8-5. Dialogue Message Flow

The dialogue performance will be discussed for each particular case. The test bed for case one is illustrated in Figure 8-6. The call setup performance and the voice quality measurements for eight test runs are shown in Table 8-5 and Table 8-6.

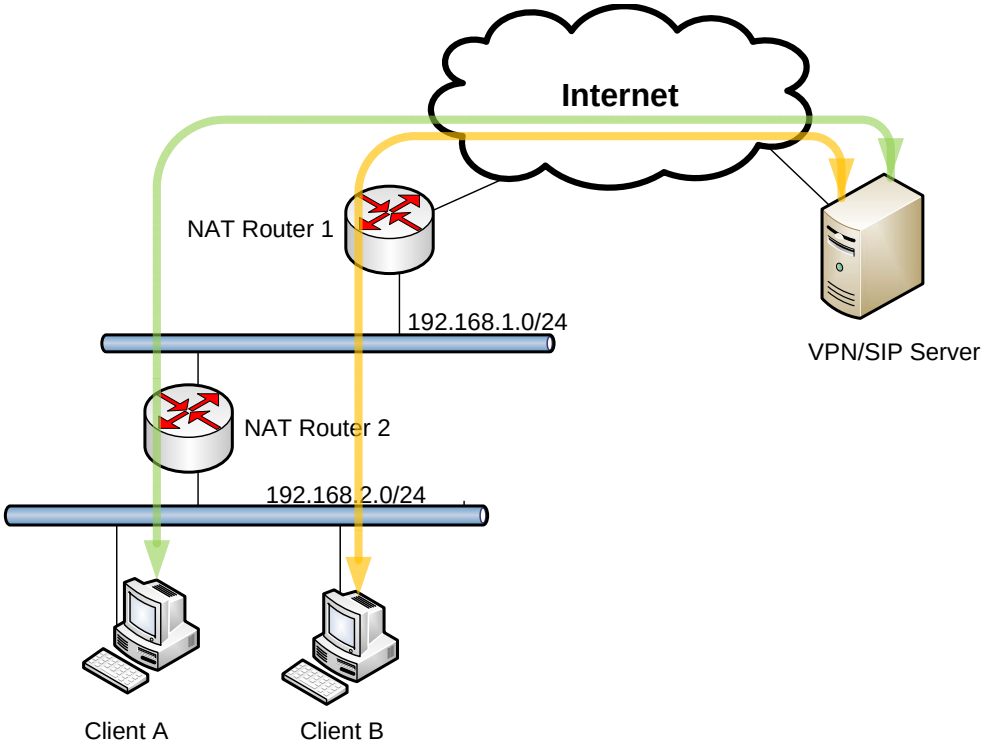


Figure 8-6. Test bed for case one

Table 8-5. VoIP call setup measurement for case one

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
a1(ms)	15.6	17.2	15.3	17.9	16.3	18.4	16.3	17.2	16.8	1.1
a2(ms)	8.1	10.5	9.4	9.5	9.1	6.9	7.2	7.6	8.5	1.3
c2(ms)	14.2	15.9	14.7	15.3	15.7	12.2	14.3	12.7	14.4	1.3
c5(ms)	43.9	45.3	46.2	47	45.8	44.7	43.2	48.1	45.5	1.6

Table 8-6. VoIP voice quality measurement for case one

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
Uplink RTP delay (ms)	39.7	40.4	39.9	40.5	39.6	39.8	39.9	40.1	40.0	0.3
Downlink RTP delay (ms)	39.8	40.6	39.2	40.8	40.1	39.5	40.2	41.1	40.2	0.7
Uplink RTP Jitter (ms)	6.1	4.0	4.0	4.5	5.2	4.4	6.4	5.6	5.0	0.9
Downlink RTP Jitter (ms)	4.4	4.8	4.6	4.7	4.0	4.6	5.3	5.3	4.7	0.4
Uplink Packet Loss rate	1.3%	1.5%	0.8%	1.2%	1.2%	1.6%	0.8%	0.5%	1.1%	0.004
Downlink Packet Loss rate	0.9%	0.5%	1.1%	0.9%	1.1%	1.2%	0.6%	0.4%	0.8%	0.003

The test bed for case two is shown in Figure 8-7. The call setup and voice quality performance for eight test runs are shown in Table 8-7 and Table 8-8.

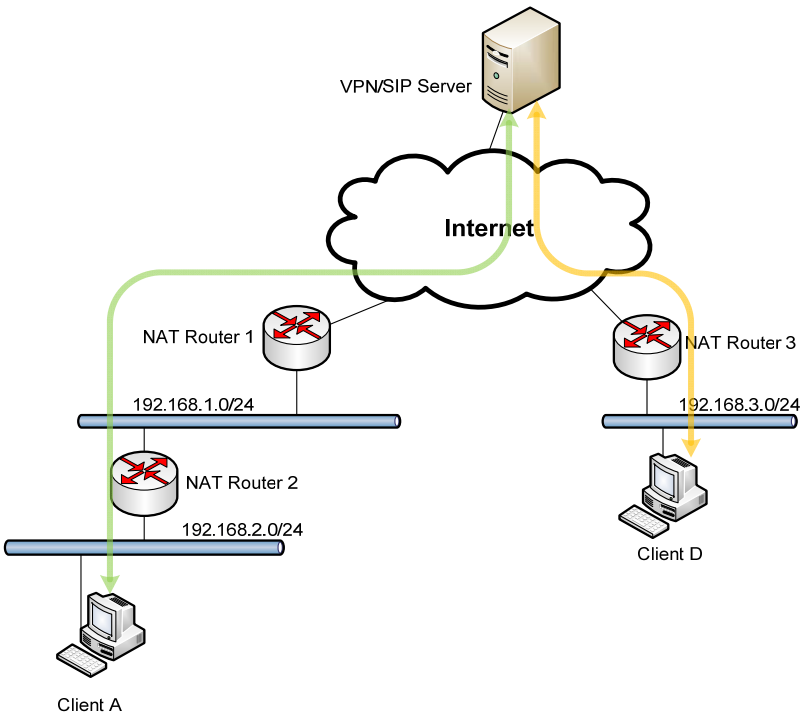


Figure 8-7. Test bed for case two

Table 8-7. VoIP call setup measurements for case two

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
a1(ms)	14.8	14.9	15.2	14.1	16.2	13.9	14.6	15.2	14.9	0.7
a2(ms)	8.5	7.2	8.4	8.2	7.8	8.7	7.9	7.9	8.1	0.5
c2(ms)	14.1	12.5	13.9	13.2	13.1	14.2	13.4	14.2	13.6	0.6
c5(ms)	42.6	43.1	41.7	43.7	42.7	41.9	43.1	42.9	42.7	0.7

Table 8-8. VoIP voice quality measurements for case two

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
Uplink RTP delay (ms)	39.6	39.9	40.1	40.2	40.2	39.7	39.9	39.8	39.9	0.2
Downlink RTP delay (ms)	39.8	40.2	40.1	39.9	40.3	39.9	40.2	40.2	40.1	0.2
Uplink RTP Jitter (ms)	5.8	3.0	4.1	4.5	5.4	3.9	6.8	4.7	4.8	1.2
Downlink RTP Jitter (ms)	9.6	7.4	6.9	8.5	9.5	6.1	7.7	8.2	8.0	1.2
Uplink Packet Loss rate	1.9%	2.4%	1.3%	1.1%	1.4%	4.1%	0.9%	0.7%	1.7%	0.011
Downlink Packet Loss rate	1.9%	0.4%	1.8%	1.7%	1.1%	1.9%	2.5%	0.9%	1.5%	0.007

The test bed for case three is shown in

Figure 8-8. The call setup and voice quality performance for eight8 test runs are shown in Table 8-9 and Table 8-10.

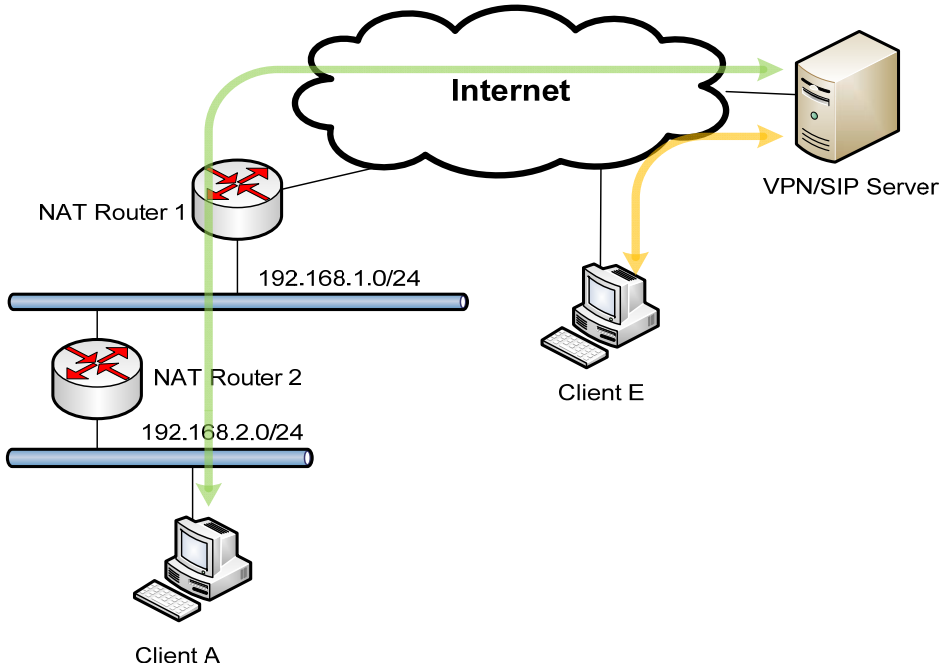


Figure 8-8. Test bed for case three

Table 8-9. VoIP call setup measurements for case three

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
a1(ms)	13.1	12.9	14.1	13.7	14.6	15.2	13.3	14.3	13.9	0.8
a2(ms)	7.2	7	8.1	7.3	7.1	8.9	7.8	7.9	7.7	0.6
c2(ms)	13.4	12.1	13.6	13.2	12.5	14.1	12.8	13.1	13.1	0.6
c5(ms)	42.1	41.9	40.9	44.1	42.1	40.8	41.3	41.9	41.9	1.0

Table 8-10. VoIP voice quality measurements for case three

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
Uplink RTP delay (ms)	39.9	40.1	39.7	39.8	40.4	39.7	39.6	40.4	40.0	0.3
Downlink RTP delay (ms)	39.7	40.1	40.2	40.1	40.0	39.9	40.1	40.3	40.1	0.2
Uplink RTP Jitter (ms)	6.9	4.7	5	8.1	4.8	9.6	7.3	4.4	6.4	1.9
Downlink RTP Jitter (ms)	8.3	5.5	5.4	7.4	5.5	10.2	8.3	5.5	7.0	1.8
Uplink Packet Loss rate	1.0%	0.8%	0.2%	0.2%	2.4%	4.5%	0.4%	0.1%	1.2%	0.015
Downlink Packet Loss rate	0.2%	0.1%	3.3%	3.3%	0.3%	0.5%	2.0%	0.7%	1.3%	0.014

The test bed for case four is illustrated in Figure 8-9. The call setup and voice quality performance for eight test runs are shown in Table 8-11 and Table 8-12.

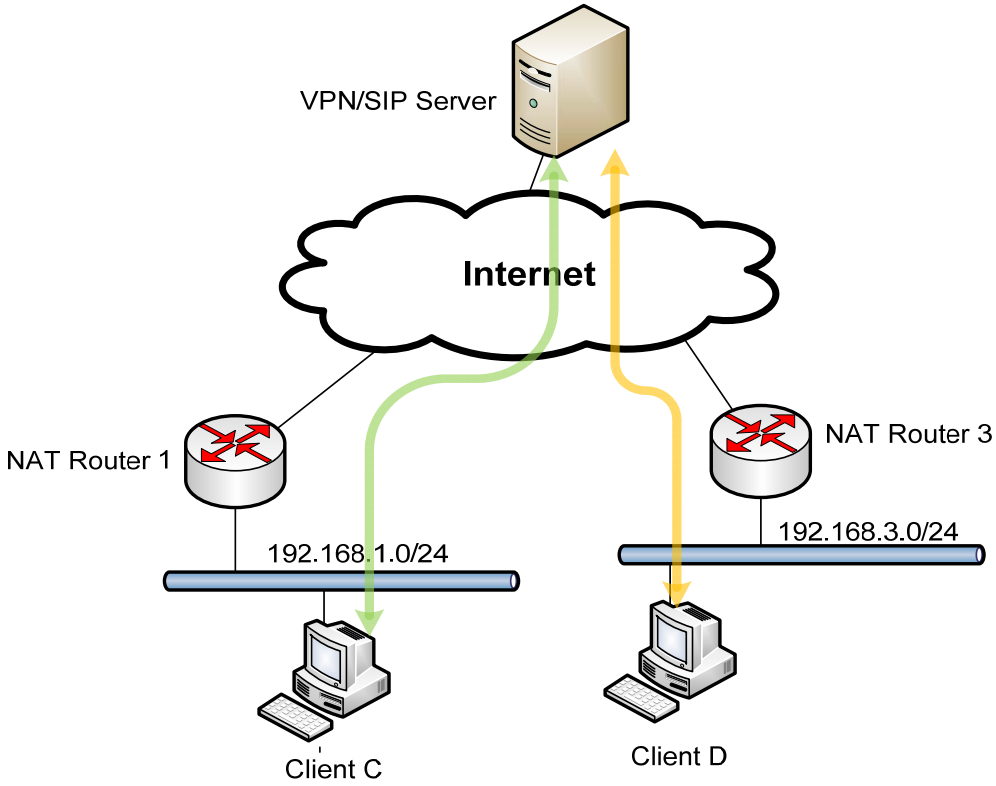


Figure 8-9. Test bed of case four

Table 8-11. VoIP call setup measurements for case four

	Run1	Run2	Run3	Run4	Run5	Run6	Run7	Run8	Mean	σ
a1(ms)	13.4	13.1	12.9	13.8	13.5	14.6	13.1	14.2	13.6	0.6
a2(ms)	7.2	6.1	7.6	6.8	6.8	6.7	6.9	8.4	7.1	0.7
c2(ms)	12.6	12.4	12.8	12.1	11.8	11.7	12.4	14.2	12.5	0.8
c5(ms)	41.6	41.9	41.6	42.6	42.7	43.8	42.1	43.2	42.4	0.8

Table 8-12. VoIP voice quality measurements for case four

	Run1	Run2	Run3	Run4	Run5	Run6	Run7	Run8	Mean	σ
Uplink RTP delay (ms)	39.6	39.8	39.8	39.6	40.0	39.3	39.7	40.3	39.8	0.3
Downlink RTP delay (ms)	39.8	40.7	40.2	40.2	40.2	39.9	39.1	40.6	40.1	0.5
Uplink RTP Jitter (ms)	3.9	8.6	5.4	6.6	9.1	4.7	5.3	6.4	6.3	1.8
Downlink RTP Jitter (ms)	8.1	5.6	7.1	3.6	4.2	9.2	8.1	6.9	6.6	2.0
Uplink Packet Loss rate	1.2%	0.8%	3.4%	0.5%	2.1%	0.2%	0.3%	3.9%	2%	0.014
Downlink Packet Loss rate	0.8%	2.1%	1.2%	0.3%	2.3%	1.0%	1.8%	0.3%	1%	0.008

The test bed for case five is represented in Figure 8-10. The call setup and voice quality performance for eight test runs are shown in Table 8-13 and Table 8-14.

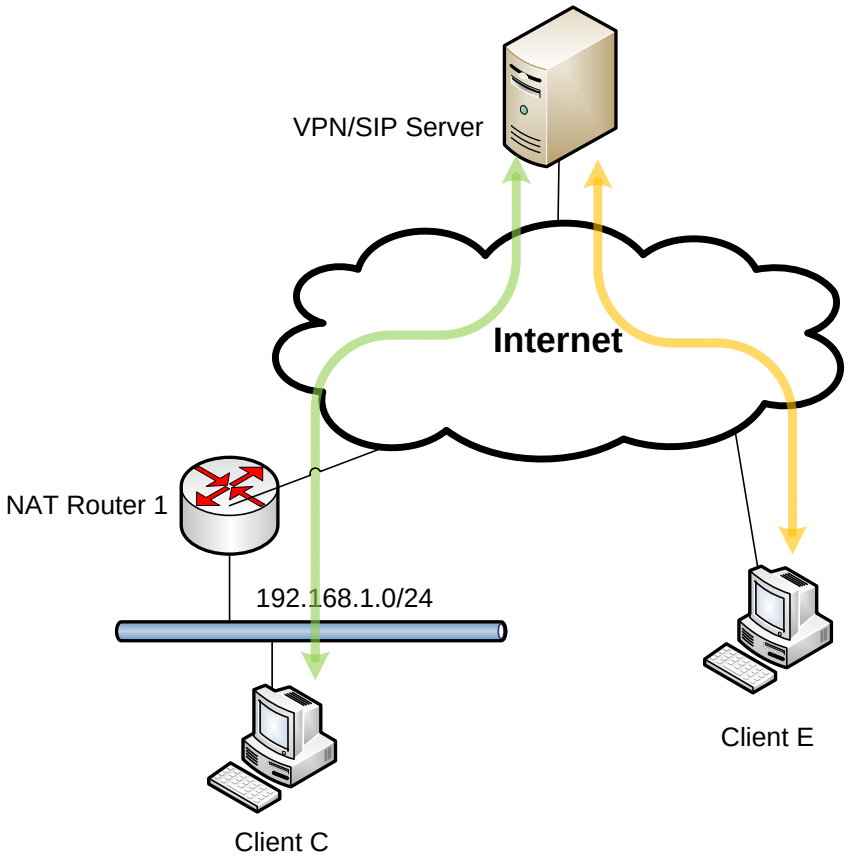


Figure 8-10. Test bed of case five

Table 8-13. VoIP call setup measurements for case five

	Run1	Run2	Run3	Run4	Run5	Run6	Run7	Run8	Mean	σ
a1(ms)	12.9	12.4	12.7	11.9	12.1	12.9	13.5	12.6	12.6	0.5
a2(ms)	7	5.9	6.7	5.8	6.6	6.3	6.8	6.2	6.4	0.4
c2(ms)	12.2	11.4	11.6	10.7	11.9	11.5	11.8	11.4	11.6	0.4
c5(ms)	39.2	37.7	38.9	39.5	39.1	38.2	38.8	39.2	38.8	0.6

Table 8-14. VoIP voice quality measurements for case five

	Run1	Run2	Run3	Run4	Run5	Run6	Run7	Run8	Mean	σ
Uplink RTP delay (ms)	39.8	40.1	40.0	39.9	40.3	39.5	39.9	39.7	39.9	0.2
Downlink RTP delay (ms)	39.6	39.7	40.1	40.3	40.0	39.7	39.9	40.4	40.0	0.3
Uplink RTP Jitter (ms)	10.7	7.4	8.9	4.2	3.8	6	9.6	5.5	7.0	2.6
Downlink RTP Jitter (ms)	9.5	12.1	11.6	9.7	3.9	4.3	10.9	9.9	9.0	3.2
Uplink Packet Loss rate	3.9%	4.1%	0.4%	0.1%	0.4%	0.7%	0.2%	0.1%	1.2%	0.017
Downlink Packet Loss rate	0.3%	0.2%	0.1%	0.1%	0.2%	0.1%	0.1%	0.1%	0.2%	0.001

Figure 8-11 shows the test bed for case six. The VoIP call setup and voice quality performance for eight test runs are shown in Table 8-15 and Table 8-16.

.

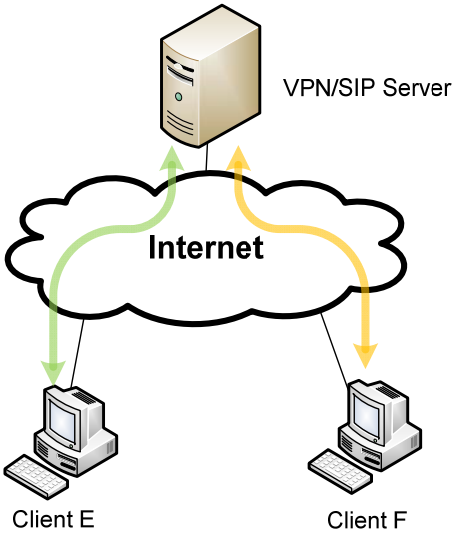


Figure 8-11. Test bed of case six

Table 8-15. VoIP call setup measurements for case six

	Run1	Run2	Run3	Run4	Run5	Run6	Run7	Run8	Mean	σ
a1(ms)	12.4	12.1	12.3	12.5	13	11.9	12.8	12.4	12.4	0.4
a2(ms)	6.8	6.7	5.9	6.4	3.9	6	5.9	6.4	6.0	0.9
c2(ms)	11.2	11.3	11.1	11.4	8.5	11.3	10.6	11.2	10.8	1.0
c5(ms)	37.9	37.4	39.1	38.1	37.1	39.2	38.7	38.4	38.2	0.8

Table 8-16. VoIP voice quality measurements for case six

	Run1	Run2	Run3	Run4	Run5	Run6	Run7	Run8	Mean	σ
Uplink RTP delay (ms)	39.9	40.0	39.8	41.2	38.9	39.1	40.5	39.6	39.9	0.7
Downlink RTP delay (ms)	40.5	40.7	40.4	40.6	40.6	40.6	40.0	39.9	40.4	0.3
Uplink RTP Jitter (ms)	7.2	12.6	9.8	9.8	10.2	12.0	10.7	11.2	10.4	1.7
Downlink RTP Jitter (ms)	6.2	12.9	7.6	9.1	11.4	10.9	9.3	8.4	9.5	2.2
Uplink Packet Loss rate	0.2%	0.4%	0.1%	0.3%	0.6%	0.1%	0.2%	0.4%	0.3%	0.002
Downlink Packet Loss rate	1.1%	1.4%	2.0%	0.8%	0.4%	2.4%	0.1%	0.9%	1.1%	0.008

8.2.2.2 Performance analysis

The delay specified as a1, known as ringing delay, is the time from the VoIP caller picking up the phone and dialing the callee to the callee's phone ringing and the caller receiving the local ring tone. During this period, the INVITE and 180 Ringing messages are transmitted between caller and callee via an SIP server through the IPSec tunnel. From our test results (in the previous section), we observe that the mean of this delay for all the six tests ranged from 12.4 ms to 16.8 ms. This delay has four main components: the encryption process, forwarding process, decryption process, and SIP processing. In order to traverse the NATs between the SIP participants, the SIP clients must communicate with the server through an IPSec VPN tunnel. This requires that every packet be encrypted by ESP before sending and be decrypted at the other end of the tunnel. This encryption and decryption process must occur for all SIP participants, both clients and server. The forwarding process in the SIP server

requires decrypting the packet, examining the SIP information, making a forwarding decisions, encrypting the outgoing packet, and forwarding it to the callee according to the SIP information which the SIP server examined. During the SIP processing period, the SIP participant decrypts the ESP packet and processes the incoming SIP message.

The delay specified as a2, Caller Transmit Time, is the period from the caller receiving the 200 OK message, which indicates that the callee has picked up the phone, to the callee receiving an ACK from the caller, which completes the call establishment phase. The Corresponding Callee Transmit Time (c2) is the period of time from the callee answering the call until the end of the call establishment phase. The results of our measurements show that the average Caller Transmit Time and Callee Transmit Time are only 6-8.5 ms and 12.8-14.4 ms, respectively.

The delay specified as c5, Termination Delay, is the time taken to terminate the SIP session. In our experiments, we assume that the callee disconnects the call first, thus the Callee hangs up and generates a BYE message. The caller confirms the BYE message with a 200 OK response. According to our measurements, this period of time is 38.2 ms to 45.5 ms.

Table 8-17: Mean VoIP call setup measurements for all six cases

	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**	**Mean**	**σ**
a1(ms)	16.8	14.9	13.9	13.6	12.6	12.4	14.0	1.63
a2(ms)	8.5	8.1	7.7	7.1	6.4	6.0	7.3	0.98
c2(ms)	14.4	13.6	13.1	12.5	11.6	10.8	12.7	1.32
c5(ms)	45.5	42.7	41.9	42.4	38.8	38.2	41.6	2.70

The media quality of VoIP is one of the most important things to be taken into consideration for the perceived quality of Internet telephony. In this paper, measured latency, jitter, and packet loss of a session in order to evaluate the performance of the system for media traffic. The results from Table 8-6, Table 8-8, Table 8-10, Table 8-12, Table 8-14, and Table 8-16 are summarized in Table 8-18.

Table 8-18: Means from the six cases

	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**	**Mean**	**σ**
Uplink RTP delay (ms)	40.0	39.9	40.0	39.8	39.9	39.8	39.9	0.09
Downlink RTP delay (ms)	40.2	40.1	40.1	40.1	40.0	39.5	40	0.25
Uplink RTP Jitter (ms)	5.0	4.8	6.4	6.3	7.0	4.4	5.6	1.05
Downlink RTP Jitter (ms)	4.7	8.0	7.0	6.6	9.0	4.6	6.6	1.76
Uplink Packet Loss rate	1.1%	1.7%	1.2%	2%	1.2%	1.6%	1.5%	0.36%
Downlink Packet Loss rate	0.8%	1.5%	1.3%	1%	0.2%	1.2%	1.0%	0.46%

The average Latency for both uplink and downlink (a4 and c3) is around 40 ms for all tested cases. As the total end-to-end latency should be less than 180 ms[40], these two delays are significant, but the total end-to-end latency delay should still be acceptable to a human user. As the distance between the SIP clients and server is different in each situation, a long path delay will further add to the delay resulting in a degradation of the perceived media quality.

The jitter (variation in latency) measured in our tests ranges from 5 ms to 10 ms. Excessive jitter causes the voice to sound garbled. A de-jitter buffer is used by the receiver to compensate for jitter. To do this the receiver receives and stores packets into this buffer, then delivers samples as a constant rate stream to the playout (audio) device. The size of the de-jitter buffer can be modified to trade off the ability to hide jitter at the cost of increased end-to-end delay. A larger de-jitter buffer also leads to lower packet loss, but increased end-to-end delay. Conversely, a reduction in the size of the de-jitter

buffer decreases the end-to-end latency, but increases the amount of packet loss. A typical de-jitter buffer is designed to hold around 20 ms of samples (roughly one audio frame). Based upon out measurements a de-jitter buffer that can hide 20 ms of jitter is more than capable of hiding the jitter that we observed in our measurements.

The observed packet loss rate in all of our tests ranged from 0.2% to 2%. Packets lost during transmission include those that do not arrive before their playout time, these packets will be ignored and considered the same as lost packets. A packet loss rate exceeding 5% (empirical value) makes the voice difficult to understand for users[41]. While packet loss could affect the perceived media quality of the VOIPSec solution the packet loss rate that we observed is well below the point that the perceived voice would be difficult to understand. Note that with forward error correction techniques it is possible to decrease the impact of randomly lost single packets, at the cost of increasing the payload size.

For comparison the delay, jitter, and packet loss for sending a continuous stream of UDP packets with an inter-packet spacing of 20 ms between the source and destination for each of the six cases are shown in Table 8-19. Note that the packet sizes are the same size as the VPN encapsulated packets, thus the processing time at each NAT and the network transmission times should be the same as for the tunneled traffic. Comparing these values with the values shown in Table 8-18 we can observe that the end-to-end latency of UDP traffic is relative small. The UDP delays measured in this test indicates that compared to the time taken for encrypting and decrypting process, the processing time at NAT(s) and the network transmission times for VPN encapsulated packets are much less. Moreover, the small jitter of UDP transmission indicates the latency of UDP traffic is stable and with a small deviation from the mean latency. The small packet loss rate also indicates the NAT traverse and network transmission performance of this solution is stable, reliable and precision.

Table 8-19: Means from the six cases for UDP packets – without any VPN processing

	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**	**Mean**	**σ**
Uplink UDP delay (ms)	6.5	5.3	5.0	4.1	3.7	3.4	4.7	1.1
Downlink UDP delay (ms)	6.4	5.4	4.9	4.1	3.8	3.3	4.7	1.0
Uplink UDP Jitter (ms)	1.0	1.1	0.8	0.6	0.6	0.6	0.8	0.2
Downlink UDP Jitter (ms)	1.1	0.9	0.7	0.8	0.6	0.5	0.8	0.2
Uplink Packet Loss rate	0.1%	0.0%	0.2%	0.3%	0.1%	0.2%	0.2%	0.10%
Downlink Packet Loss rate	0.3%	0.1%	0.0%	0.1%	0.5%	0.0%	0.2%	0.20%

8.3 Extended VOIPSec solution

As mentioned earlier, IPSec provides secure data transfers on the network layer. In order to provide end–to-end protection of real time traffic, we propose to use SRTP to extend the security of the VOIPSec solution to an application to application protection of the media streams. In order to determine if this is an acceptable extension, we need to measure its performance. The test bed used was the same as used in the previous measurements, i.e., the test bed illustrated in Figure 8-1 on page 28.

Section 8.3.1 presents how SRTP works with the VOIPSec solution and what aspects will be evaluated. Section 8.3.2 presents the details of measurements and analyzes the results of the performance during a dialogue.

8.3.1 Performance of NAT traversal

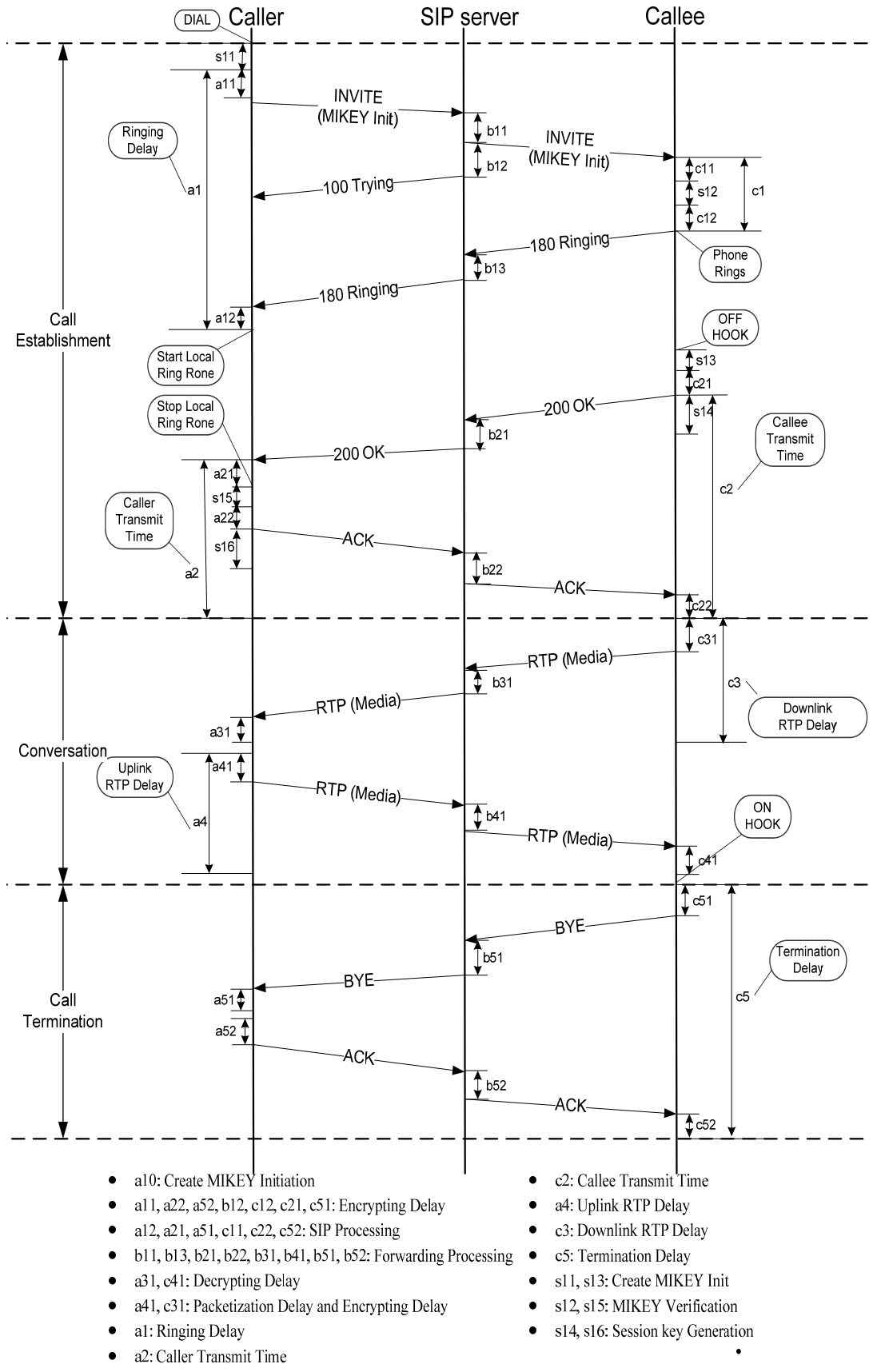
As the extended VOIPSec solution has the same NAT traversal properties of the VOIPSec solution there is no need for a separate evaluation of its performance. However, the dialogue performance **is** expected to differ, this is described in the next subsection.

8.3.2 Performance of SRTP Dialogue

In addition to the processing required for VOIPSec, SRTP requires a master key so it can derive session keys for secure transmission based on this master key. MIKEY will be used as our key management protocol. MIKEY messages are carried in SDP key management attributes ("key-mgmt" attributes). These messages are used to negotiate and to generate session keys between caller and callee during call establishment. In this experiment, we consider the case of a pre-shared key because it is a simple mechanism and causes minimal delay. The Diffie-Hellman and public key mechanisms result in a SIP message that is larger than 1300 bytes, which is not suitable for UDP transport.[7] The relative timing of pre-shared key and Diffie-Hellman for minsip were report in [42].

In the call establishment process, the caller sends a MIKEY Initiation (MIKEY init) message in the INVITE request to the caller. This MIKEY init message includes the caller's digital signature. The time taken to create the MIKEY initiation message is s11. The callee authenticates the caller before his phone rings. A MIKEY Reply is sent back along with a 200 OK message after the callee picks up the phone. The callee's digital signature is contained in the MIKEY Reply. After caller and callee verify the each other's signatures, they generate the session key which is used to encrypt the subsequent media streams based on the previous MIKEY message exchange. The time required to create the respective session keys are s13 and s14, as shown in Figure 8-12.

In Section 8.2.2, we divided the measurement into six different test cases. We will measure the performance of the extended VOIPSec solution based on these same six cases. The test bed for each of these test cases were shown in Figure 8-6 to Figure 8-11. The measurements of the VoIP call setup process for the extended VOIPSec solution are presented in Table 8-20 to Table 8-25.



- a10: Create MIKEY Initiation
- a11, a22, a52, b12, c12, c21, c51: Encrypting Delay
- a12, a21, a51, c11, c22, c52: SIP Processing
- b11, b13, b21, b22, b31, b41, b51, b52: Forwarding Processing
- a31, c41: Decrypting Delay
- a41, c31: Packetization Delay and Encrypting Delay
- a1: Ringing Delay
- a2: Caller Transmit Time
- c2: Callee Transmit Time
- a4: Uplink RTP Delay
- c3: Downlink RTP Delay
- c5: Termination Delay
- s11, s13: Create MIKEY Init
- s12, s15: MIKEY Verification
- s14, s16: Session key Generation

Figure 8-12. Dialogue message flow of application level VOIPSec solution[43]

As with the earlier testing, the results are based upon eight runs of the same test case. Table 8-20 shows the measurements of the VOIPsec call set up for case one. The subsequent tables show the other five test cases.

Table 8-20. Extended VOIPSec call set up measurement for case one

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
a1(ms)	33.9	32.7	35.1	32.5	31.8	33.9	32.6	32.4	33.1	1.1
a2(ms)	18.4	17.9	18.5	19.3	17.8	18.3	17.9	19.4	18.4	0.6
c2(ms)	31.1	29.6	29.5	31.40	28.7	30.6	29.7	30.2	30.1	0.9
c5(ms)	46.5	47.3	45.9	47.1	46.3	48.1	47.8	49.2	47.3	1.1

Table 8-21. Extended VOIPSec call set up measurement for case two

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
a1(ms)	29.7	30.5	29.9	29.3	29.6	29.1	32.1	30.2	30.1	0.9
a2(ms)	17.4	16.8	17.1	17.3	16.9	17.4	16.8	16.9	17.1	0.3
c2(ms)	29.1	27.8	28.9	29.5	28.4	29.3	27.1	28.7	28.6	0.8
c5(ms)	43.5	43.6	43.8	42.9	44.1	45.1	43.2	42.8	43.6	0.7

Table 8-22. Extended VOIPSec call set up measurement for case three

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
a1(ms)	29.6	29.5	31.1	27.3	28.4	29.5	28.5	29	29.1	1.1
a2(ms)	14.9	14.7	16.1	14.9	17.4	14.5	16.2	15.3	15.5	1.0
c2(ms)	25.4	26.7	29.1	24.5	34.7	25.1	26.1	27.5	27.4	3.3
c5(ms)	42.1	42.8	41.6	41.4	45.2	40.6	40.3	42	42.0	1.5

Table 8-23. Extended VOIPSec call set up measurement for case four

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
a1(ms)	29.8	26.4	27.5	28.1	29.4	29.6	29.8	28.1	28.6	1.3
a2(ms)	15.4	13	14.3	14.5	14.7	14.2	14.8	13.9	14.4	0.7
c2(ms)	27.9	25.7	24.1	27.8	26.1	26.5	28.3	26.8	26.7	1.4
c5(ms)	40.7	42.5	42.1	41.8	40.8	39.2	42.8	43.9	41.7	1.5

Table 8-24. Extended VOIPSec call set up measurement for case five

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
a1(ms)	27.8	27.1	27.5	27.6	27.6	27.8	30.1	27.5	27.9	0.9
a2(ms)	11.5	13.6	13.3	13.1	15.1	13.1	13.2	14.1	13.4	1.0
c2(ms)	24.1	22.2	26.9	24.1	26.8	30.2	25.1	27.2	25.8	2.5
c5(ms)	38.7	40	41.6	42.3	37.9	39.7	43.7	38.5	40.3	2.0

Table 8-25. Extended VOIPSec call set up measurement for case six

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
a1(ms)	25.30	29.1	26.7	27.9	25.9	26.3	27.5	27.1	27.0	1.2
a2(ms)	14.20	12.3	13.8	16.1	10.3	14.5	12.8	12	13.3	1.8
c2(ms)	28.20	23.9	24.1	29.1	19.1	25.7	23.9	24.1	24.8	3.1
c5(ms)	39.80	37.1	40.1	36.2	38.1	39.7	38.4	39	38.6	1.4

Table 8-26 to Table 8-31 provide the media performance of the extended VOIPSec solution for each of the six test cases. For each of the test cases eight calls were made between the caller and the callee.

Table 8-26. Media quality measurement of extended VOIPSec in case one

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
Uplink SRTP delay (ms)	40.3	40.2	40.2	40.1	40.1	40.3	40.1	40.2	40.2	0.1
Downlink SRTP delay (ms)	40.1	40.2	39.9	40	40.1	40.2	40.1	40.3	40.1	0.1
Uplink SRTP Jitter (ms)	9.8	9.4	11.8	10.6	9.4	8.2	9.4	7.3	9.5	1.4
Downlink SRTP Jitter (ms)	9.4	8.6	10.3	10.4	9.3	12.5	6.1	8.4	9.4	1.8
Uplink Packet Loss rate	0.10%	0.50%	0.90%	0.70%	1.50%	3.20%	0.70%	2.10%	1.2%	0.010
Downlink Packet Loss rate	1.10%	3.80%	0.90%	0.40%	0.60%	0.50%	1.10%	2.50%	1.4%	0.012

Table 8-27. Media quality measurement of extended VOIPSec in case two

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
Uplink SRTP delay (ms)	40	40.1	40.2	39.9	39.8	40.2	40.1	40.1	40.1	0.1
Downlink SRTP delay (ms)	40.2	40.1	39.9	39.8	40	40.1	40.3	40.2	40.1	0.2
Uplink SRTP Jitter (ms)	7.9	9.5	9.3	8.5	9.1	9.4	4.9	8.3	8.4	1.5
Downlink SRTP Jitter (ms)	9.3	7.3	6.1	8.8	7.3	6.3	10.4	9.5	8.1	1.6
Uplink Packet Loss rate	0.10%	0.40%	0.80%	0.20%	0.50%	1.60%	0.80%	3.80%	1.0%	0.012
Downlink Packet Loss rate	0.90%	2.30%	0.30%	0.40%	0.10%	0.90%	0.60%	1.30%	0.9%	0.007

Table 8-28. Media quality measurement of extended VOIPSec in case three

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
Uplink SRTP delay (ms)	40.1	40.2	40.1	39.9	40.2	40.3	40.1	40.1	40.1	0.1
Downlink SRTP delay (ms)	40	40.1	40.2	39.8	40	40.1	40	39.9	40.0	0.1
Uplink SRTP Jitter (ms)	7.7	8.3	7.4	9.4	6.8	9.4	10.2	6.5	8.2	1.3
Downlink SRTP Jitter (ms)	8.6	7.2	9.4	9.8	8.9	4.7	8.7	8.5	8.2	1.6
Uplink Packet Loss rate	0.60%	0.20%	0.10%	0.80%	2.50%	1.10%	0.20%	0.50%	0.8%	0.008
Downlink Packet Loss rate	0.90%	1.10%	0.40%	0.50%	1.10%	0.60%	0.80%	0.10%	0.7%	0.004

Table 8-29. Media quality measurement of extended VOIPSec in case four

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
Uplink SRTP delay (ms)	40.1	40.2	39.9	40	39.8	40.1	40.2	40	40.0	0.1
Downlink SRTP delay (ms)	40	40	39.9	39.8	40.1	39.9	39.9	40	40.0	0.1
Uplink SRTP Jitter (ms)	6.1	5.9	8.7	9.6	7.8	9	7.4	8.8	7.9	1.4
Downlink SRTP Jitter (ms)	8.1	9.3	7.2	8.5	9.7	5.8	9	8.9	8.3	1.3
Uplink Packet Loss rate	0.20%	0.50%	0.30%	0.60%	0.40%	0.80%	1.40%	3.00%	0.9%	0.009
Downlink Packet Loss rate	0.10%	0.90%	0.50%	3.00%	0.90%	0.70%	1.50%	1.00%	1.1%	0.009

Table 8-30. Media quality measurement of extended VOIPSec in case five

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
Uplink SRTP delay (ms)	40.1	40.2	39.9	40	40.1	40.2	40.1	40	40.1	0.1
Downlink SRTP delay (ms)	40.2	39.8	39.9	40.1	40	40.1	39.9	39.9	40.0	0.1
Uplink SRTP Jitter (ms)	8.3	6.7	9.1	10	8.6	9.3	9.7	8.1	8.7	1.1
Downlink SRTP Jitter (ms)	8.2	7.9	7.4	8.3	8.5	7.8	8,5	7.8	8.0	0.4
Uplink Packet Loss rate	1.0%	0.7%	0.3%	0.6%	0.8%	0.2%	0.5%	1.2%	0.7%	0.003
Downlink Packet Loss rate	0.4%	0.1%	0.1%	0.3%	0.9%	0.7%	0.8%	0.4%	0.5%	0.003

Table 8-31. Media quality measurement of extended VOIPSec in case six

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Mean**	**σ**
Uplink SRTP delay (ms)	40.1	39.7	40.2	39.2	40.1	40.2	40.2	40.3	40.0	0.4
Downlink SRTP delay (ms)	40.1	39.9	40.2	39.9	40.3	40.2	40.3	39.9	40.1	0.2
Uplink SRTP Jitter (ms)	6.0	7.1	6.9	7.2	7.3	6.6	7.5	7.1	7.0	0.5
Downlink SRTP Jitter (ms)	8.1	8.3	7.2	8.5	6.5	8.3	6.6	7.8	7.7	0.8
Uplink Packet Loss rate	1.1%	0.8%	0.9%	1.3%	0.3%	0.2%	0.6%	0.1%	0.7%	0.004
Downlink Packet Loss rate	0.5%	0.2%	0.3%	1.0%	0.8%	0.3%	0.2%	1.0%	0.5%	0.003

8.3.2.1 Performance analysis

Table 8-32 summarizes the measurements of the delays during the call setup and termination processes of extended VOIPSec for all test cases. While Table 8-33 compares the means of VOIPSec with the means of Extended VOIPSec.

Table 8-32: Mean delays during the call setup & termination of extended VOIPSec for all six cases

	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**	**Mean**	**σ**
a1(ms)	33.1	30.1	29.1	28.6	27.9	26.3	27.0	2.30
a2(ms)	18.4	17.1	15.5	14.4	13.4	14.5	13.3	1.87
c2(ms)	30.1	28.6	27.4	26.7	25.8	25.7	24.8	1.71
c5(ms)	47.3	43.6	42.0	41.7	40.3	39.7	38.6	2.75

Table 8-33:Mean of delays during the call setup and termination for VOIPSec and Extended VOIPSec

	VOIPSec	**Extended VOIPSec**
a1(ms)	14.0	27.0
a2(ms)	7.3	13.3
c2(ms)	12.7	24.8
c5(ms)	41.6	38.6

The ringing delay specified as a1 is the time from the caller inviting the callee to a conversation to the caller receiving the local ring tone. From the measurements for all the test cases, we can observe that this delay ranges from 27 ms to 33 ms (with a mean of 27 ms). Compared to 12-16 ms, with a mean of 14 ms, for the corresponding delay of the VOIPSec solution, the ringing delay of the

extended VOIPSec solution is a substantial increase (approximately double the time). (See the above tables.) The main reason for this increase is the increased transmission delay of the INVITE message, which includes the fact that the MIKEY Init message cannot be ignored. The MIKEY Init message contains the caller's digital signature, which is used to authenticate the caller. This MIKEY Init message is included in the SIP message as an SDP attribute which causes 445 additional bytes to be added to the SIP message. Another element of this increased delay is the time taken to verify the MIKEY signature, specified as s12. The callee uses the shared key to decrypt and check the caller's identification.

Similar to the ringing delay, the caller transmit time (a2) and callee transmit time (c2) are longer as compared to the corresponding time in the VOIPSec solution. An explanation could be that the 200 OK message carries an additional MIKEY Reply that contains the callee's identification. The mutual authentication between caller and callee is completed after the caller receives this 200 OK message. The caller needs time to verify the callee's identification; this time is specified as s15. After this the caller and callee independently generate an encrypt key for the subsequent media streams; these delays are specified as s14 and s16, respectively.

Comparing the message flows in the two solutions, indicated in Figure 8-5and Figure 8-12, we can observe that the termination processes of these two solutions are basically the same. From the measurements, the termination delay times (c5) in the extended VOIPSec solution and in the VOIPSec system are comparable (as expected).

The time taken to create the MIKEY Init (s11) and MIKEY Reply (s13) were not measured in this experiment due to our focus on the performance of the VOIPSec solutions in different network infrastructures instead of the user agent's performance. The performance of the user agent is discussed and measured in detail in [43].

The media quality test results, from Table 8-26 to Table 8-31, are summarized in Table 8-34.The average transmit latency of the SRTP packets for both incoming streams and outgoing streams of the complete conversation were around 40 ms for all test cases. These results indicate that the additional latency of SRTP streams has no significant effect on the latency of ordinary RTP streams. In other words, SRTP does not cause extra delay for the media transmissions as compared to the regular RTP transmission. Therefore, the added benefits of SRTP do not incur a significant cost in terms of end-to-end delay.

Table 8-34: Summary of the mean of the media quality measurement of extended VOIPSec from all six cases

	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**	**Mean**	**σ**
Uplink SRTP delay (ms)	40.2	40.1	40.1	40.0	40.1	40.0	40.1	0.075
Downlink SRTP delay (ms)	40.1	40.1	39.9	40.0	40.0	40.1	40.0	0.082
Uplink SRTP Jitter (ms)	9.5	8.4	6.5	7.9	8.7	7.0	8.0	1.110
Downlink SRTP Jitter (ms)	9.4	8.1	8.5	8.3	8.0	7.7	8.3	0.589
Uplink Packet Loss rate	1.2%	1.0%	0.50%	0.9%	0.7%	0.7%	0.83%	0.003
Downlink Packet Loss rate	1.4%	0.9%	0.10%	1.1%	0.5%	0.5%	0.75%	0.005

9 Conclusions and future work

This section summarizes the main conclusions of this thesis. It compares the VOIPSec solution with the extended VOIPSec solution. We make some recommendations for different types of customers. We end by suggesting some future work that may improve the performance of the VOIPSec and extended VOIPSec solution or that may improve the performance analysis of these two solutions.

9.1 Conclusions

The aim of this thesis project was to design, deploy, and evaluate a potential universal VoIP solution which could operate in most network infrastructures. The main problem currently facing increased VoIP deployment is how to help the VoIP participants successfully traverse NATs. Existing NAT traversal mechanisms each have their own limitations as described in section 3.2. Not only are NATs potential barriers to successful VoIP call establishment, but the existing NAT traversal mechanisms increase call setup delay.

In this thesis project, we designed, deployed, and evaluated a new NAT traverse solution for VoIP that utilizes an IPSec VPN (in reality many IPSec VPNs centered on our SIP/VPN server). In this approach the SIP participants route their VoIP traffic through the NAT inside a VPN. In the proposed VOIPSec solution, the IPSec VPN tunnel connects each of the SIP clients to a SIP server, thus making all of the potential SIP participants reachable. All SIP signaling and media traffic for VoIP calls are transmitted through this prior established tunnel. This VPN tunnel provides the desired universal means for VoIP traffic to traverse NAT equipment (assuming that all NATs are capable of handling IPSec VPNs correctly). Additionally, the IPSec VPN also guarantees the security of VoIP calls at the IP level.

In order to improve the security level of media streams for the VOIPSec solution, we deployed and evaluated an extended VOIPSec solution which provides end-to-end protection of the real time media traffic. In this extended VOIPSec solution, we used SRTP instead of RTP to carry the media content. This extended method was shown to provide all of the advantages of VOIPSec and SRTP without any additional delay for the media traffic (as compared to the VoIPSec solution).

One of our conclusions, based upon our work is that for VoIP calls that do not need end-to-end level security, we recommend the use of the VOIPSec solution as a means to solve the NAT traversal problem and to protect traffic at the general IP level. When application to application security is **not** needed we prefer the VOIPSec solution to the extended VOIPSec solution for the following reasons:

- The VOIPSec solution provides IP level protection which should satisfy the security requirements for most users. SRTP protection extends the VOIPSec solution and it costs some additional time to negotiate the parameters for SRTP. Our test results show that the time for call setup for the extended VOIPSec solution is twice time the time needed for the VOIPSec solution. For users who do **not** need such a high-level security, it may not be worth the longer wait to set up the call. Note that although an increase of a factor of two may seem large, in practice the total additional waiting time **may not** even be noticeable by the user.

- The extended VOIPSec solution requires the use of user agents that support SRTP. Compared to the extended VOIPSec solution, theVOIPSec solution does not require a special user agent and

all normal VoIP clients in the market are compatible with this system. It is easy and simple to promote in the market. There are relatively few VoIP applications that support SRTP, although this situation is changing. As of today, the following SIP user agents are known to support SRTP: minisip, PJSIP, Cisco's IOS as of release 12.4(15)T, Grandstream's FreeSWITCH, Asterisk, Microsoft's RTC, and CounterPath's Eyebeam. As more SIP user agents support SRTP, the extended VOIPSec solution will be available to users of these SIP user agents.

- Minisip is not stable and its development is incomplete. In particular, Minisip for the Windows XP platform is not fully tested. For the casual user minisip is currently too complicated and difficult to use as a VoIP user agent.

9.2 Future work

The proposed VOIPSec solution offers a possible mechanism to solve the NAT traversal problem in complicated network infrastructures. Based on our experimental results the performance of the VOIPSec and extended VOIPSec solutions should be acceptable for users. However, one problem should be noted. All of the test beds were set up in an ideal condition and private environments and there was not much other traffic in the test network. As the number of users and the amount of traffic increase, the latency of the VOIPSec solution may increase and the voice quality will degrade. These two solutions should be tested and evaluated in a real environment network. Additionally, testing with human testers or with automated QoS test equipment should be conducted.

In section 8.2.1.1 we assumed that the measurements were normally distributed and that the mean is representative of the delay. However, additional measurements need to be made to see if this assumption is correct. This means that more than 30 runs should be made for each scenario.

We have not compared the performances of the VOIPSec solution with different cryptographic algorithms for ESP. We believe different cryptographic algorithm may influence the QoS of the VoIP call. However, changing the algorithm used for ESP is **unlikely** to *significantly* improve the performance of the media transmission for the VOIPSec solution as the current encryption+decryption delay is already rather small..

Another possible means to improve performance of the VOIPSec solution is related to the RTP packet size. The size of the RTP packets has a large effect on their transmission latency. The RTP frame sizes are different when using different CODECs. It is unclear what the exact tradeoff is with regard to network congestion as a function of the packet size, but for a given amount of content sending more small packets is less efficient than sending fewer larger packets (due to the overhead of the IP, UDP, and RTP headers – unless header compression can be used). A larger frame size will increase the packetization delay, but could reduce the number of packets that need to be processed and will increase the efficiency of the user of the network link – while also decreasing the delay added due to NAT processing. However, measuring the tradeoff remains a future task.

Note that with forward error correction techniques it is possible to decrease the impact of randomly lost single packets, at the cost of increasing the payload size. An evaluation of adding forward error correction should be examined in conjunction with the results of the study suggested above.

References

1 Hui Min Chong; H. Scott Matthews, Comparative Analysis of Traditional Telephone and Voice-over-Internet Protocol (VoIP) Systems, Electronics and the Environment, 2004. Conference Record. 2004 IEEE International Symposium on Volume, Issue, 10-13 May 2004 Page(s): 106 - 111

2 Jennifer Sundquist and Nick Service, White Paper-Top 10 Myths about VoIP, Epygi Technologies Ltd, April 25, 2006.

3 QuickStudy: Packet-Switched vs. Circuit-Switched Networks. URL. http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=41904

4 Voice over Internet Protocol, http://www.protocols.com/pbook/VoIPFamily.htm

5 SIP and NAT - An Introduction, x-console on Ocobert 05, 2006, http://www.voipuser.org/forum_topic_7295.html

6 Session Initiation Protocol, http://www.telecomspace.com/vop-sip.html

7 J. Rosenberg, et al., SIP: Session Initiation Protocol, Internet Engineering Task Force, RFC 3261, June 2002. http://www.ietf.org/rfc/rfc3261.txt

8 G. Q. Maguire Jr., Practical Voice Over IP (VoIP): SIP and related protocols slides, Royal Institute of Technology (KTH), Spring 2008, Period 4.

9 A. Johnston, et al., Session Initiation Protocol (SIP) Basic Call Flow Examples, Network Working Group, RFC3665, December 2003. http://www.ietf.org/rfc/rfc3665.txt

10 M. Handley and V. Jacobson, SDP: Session Description Protocol, Internet Engineering Task Force, RTC 2327, April 1998. http://www.ietf.org/rfc/rfc2327.txt

11 M. Handley, et al., SDP: Session Description Protocol, Internet Engineering Task Force, RTC 4566, July 2006, http://www.ietf.org/rfc/rfc4566.txt

12 Xiaokun Yi, Adaptive Wireless Multimedia Services, Masters thesis, Royal Institute of Technology (KTH), Department of Communication systems, COS/CCS, 2006-12, May 2006

13 K. Egevang, The IP Network Address Translator (NAT), Internet Engineering Task Force, RFC 1631, May 1994, http://www.ietf.org/rfc/rfc1631.txt

14 J. Rosenberg, STUN - Simple Traversal of User Datagram Protocol (UDP)-Through Network Address Translators (NATs), Internet Engineering Task Force, RFC 3489, March 2003, http://www.ietf.org/rfc/rfc3489.txt

15 Peter Koski, Jorma Ylinen, and Pekka .Loula, The SIP-Based System Used in Connection with a Firewall. Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on.203 – 203,2006

16 J. Rosenberg, et al., SIP: Session Initiation Protocol, Internet Engineering Task Force, RFC 3261, June 2002. http://www.ietf.org/rfc/rfc3261.txt

17 M. Handley and V. Jacobson, SDP: Session Description Protocol, Internet Engineering Task Force, RTC 2327, April 1998. http://www.ietf.org/rfc/rfc2327.txt

18 J. Rosenberg, Session Traversal Utilities for NAT (STUN), Internet Engineering Task Force, RFC 5389, October 2008

19 Newport netowkrs, NAT Traversal for Multimedia over IP, 2008. http://www.newport-networks.com

20 A. La Torre Yurkov, Implementation of Traversal Using Relay Nat for SIP based VoIP, Master Thesis, Royal Institute of Technology (KTH), Stockholm, February 2006.

21 J. Rosenberg, Traversal Using Relays around NAT (TURN): Relay Extensions to Session-Traversal Utilities for NAT (STUN), TURN-11, October, 2008

22 J. Rosenberg, Interactive Connectivity Establishment， IETF Journal， Volume 2 Issue 3，

November 2006.

23 VPN Consortium, VPN Technologies: Definitions and Requirements, July 2008, http://www.vpnc.org/vpn-technologies.html

24 Virtual private network, http://en.wikipedia.org/wiki/VPN#Categorizing_VPN_security_models

25 S. Kent, et al., Security Architecture for the Internet Protocol, RFC 4301, December 2005

26 Naganand Doraswamy and Dan Harkins, IPSec: The New Security Standard for the Inter- net, Intranets, and Virtual Private Networks, 2003

27 S. Kent, et al., IP Authentication Header, Internet Engineering Task Force, RFC 4302, December 2005

28 S. Kent, et al., IP Encapsulating Security Payload (ESP), Internet Engineering Task Force, RFC 4303, December 2005

29 V. Manral, et al., Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH), Internet Engineering Task Force, RFC 4305, April 2007

30 T. Dierks, et al., The Transport Layer Security (TLS) Protocol-Version 1.2, Internet Engineering Task Force, RFC 5246, August 2008

31 C. Kaufman, et al, The Internet Key Exchange (IKE), Internet Engineering Task Force, RFC 4306, December 2005

32 D. Maughan, et al., Internet Security Association and Key Management Protocol (ISAKMP), Internet Engineering Task Force, RFC 2408, November 1998

33 D. Richard Kuhn, Thomas J. Walsh, and Steffen Fries, Security Consideration for Voice Over IP Systems, Community Contributions, June 2005

34 Newport Networks, White Paper - IPsec in VoIP Networks, http://www.newport-networks.com/whitepapers/IPsec-1.html

35 A. Huttunen, et al., UDP Encapsulation of IPSec ESP Packets, Internet Engineering Task Force, RFC 3948, January 2005

36 M. Baugher, et al., The Secure Real-time Transport Protocol (SRTP), Internet Engineering Task Force, RFC 3711, March 2004

37 H. Krawczyk，et al., HMAC: Keyed-Hashing for Message Authentication, Internet Engineering Task Force, RFC 2104, February 1997

38 J. Arkko, et al., MIKEY: Multimedia Internet KEYing, Internet Engineering Task Force, RFC 3830, August 2004

39 Markus Hidell, Internet Security and Privacy: IPSec: IKE SLIDES, Royal Institute of Technology (KTH), Autumn 2008.

40 Cole, R.G. and J. Rosenbluth, Voice Over IP Performance Monitoring, Journal of Computer Communications Review, vol. 4, no. 3, April 2001.

41 David Jacobs， How QoS appliances subdue VoIP bugbears， Oct.2005，

http://searchunifiedcommunications.techtarget.com/tip/0,289483,sid186_gci1135176_mem1,00.html

42 J. Bilien, E. Eliasson, J-O Vatn, Call establishment delay for secure VoIP, WiOpt'04, Cambridge UK, March 2004. http://www.minisip.org/publications/secvoip.pdf

43 Johan Bilien, Erik Eliasson, Joakim Orrblad, J-O Vatn, "Secure VoIP: call establishment and media protection", 2nd Workshop on Securing Voice over IP, Washington DC, June 2005

TRITA-ICT-EX-2009:105

www.kth.se