PERFORMANCE EVALUATION OF SECURE VOICEMAIL ON ASTERISK VIA SPECSIP BENCHMARK

CHALEARMPOP ONGSIT

A RESEARCH PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE (COMPUTER SCIENCE) FACULTY OF GRADUATE STUDIES MAHIDOL UNIVERSITY 2010

Research Project
entitled

PERFORMANCE EVALUATION OF SECURE VOICEMAIL ON ASTERISK VIA SPECSIP BENCHMARK

……
Mr. Chalearmpop Ongsit
Candidate

……
Lect. Sudsanguan Ngamsuriyaroj,
Ph.D.
Major advisor

……
Asst. Prof. Vasaka Visoottiviseth,
Ph.D.
Co-advisor

……
Lect. Soranun Jiwasurat,
Ph.D.
Co-advisor

……
Prof. Banchong Mahaisavariya,
M.D., Dip Thai Board of Orthopedics
Dean
Faculty of Graduate Studies
Mahidol University

……
Assoc. Prof. Supachai Tangwongsan,
Ph.D.
Program Director
Master of Science Program
in Computer Science
Faculty of Information and
Communication Technology
Mahidol University

Research Project
entitled

PERFORMANCE EVALUATION OF SECURE VOICEMAIL ON ASTERISK VIA SPECSIP BENCHMARK

was submitted to the Faculty of Graduate Studies, Mahidol University
for the degree of Master of Science (Computer Science)
on
May 14, 2010

……
Mr. Chalearmpop Ongsit
Candidate

……
Lect. Chalermpol Charnsripinyo,
Ph.D.
Chair

……
Lect. Sudsanguan Ngamsuriyaroj,
Ph.D.
Member

……
Asst. Prof. Vasaka Visoottiviseth,
Ph.D.
Member

……
Lect. Soranun Jiwasurat,
Ph.D.
Member

……
Prof. Banchong Mahaisavariya,
M.D., Dip Thai Board of Orthopedics
Dean
Faculty of Graduate Studies
Mahidol University

……
Assoc. Prof. Jarernsri L. Mitrpanont,
Ph.D.
Acting Dean
Faculty of Information and
Communication Technology
Mahidol University

ACKNOWLEDGEMENTS

I would like to especially thank to my advisor, Dr. Sudsanguan Ngamsuriyaroj, who always supports and encourages me for my entire research project. Her kindness and attention help keep me on the right way and success. I have learnt a lot from her valuable suggestions and guidance during my research. Furthermore, I have improved the systematic thinking under her supervision.

I am also very grateful to Dr. Vasaka Visoottiviseth and Dr. Soranun Jiwasurat, my co-advisors, for their invaluable suggestions throughout the research proposal. I am thankful to Dr. Chalermpol Charnsripinyo whose comments during the defense of my research project have been particularly well observed and supportive.

My appreciation also goes to all the lecturers in the Faculty of Information and Communication Technology for giving me valuable knowledge, and of course, many thanks also go to the officers who had been very helpful during this research.

I am very grateful to the Revenue Department of Thailand for providing me a scholarship and an opportunity to study here, and for the understanding and the encouragement throughout the course of my study. I would like to thank my colleagues for their help and support as well.

Finally, I am most grateful to my family for their concern, care, and love. They always encourage and keep me walking to fight on all my troubles.

Any benefits or usefulness that may be derived from my research project, I dedicate to my father, mother and all the teachers who have taught me since childhood.

Chalearmpop Ongsit

PERFORMANCE EVALUATION OF SECURE VOICEMAIL ON ASTERISK VIA SPECSIP BENCHMARK

CHALEARMPOP ONGSIT 4836584 ITCS/M

M.Sc.(COMPUTER SCIENCE)

RESEARCH PROJECT ADVISORY COMMITTEE: SUDSANGUAN NGAMSURIYAROJ, Ph.D., VASAKA VISOOTTIVISETH, Ph.D., SORANUN JIWASURAT, Ph.D.

ABSTRACT

Voice over Internet Protocol (VoIP) has become a worldwide technology, and Session Initiation Protocol (SIP) is popularly used in various implementations especially in open source software such as Asterisk. It facilitates telephony services including voicemail applications. The main concern has been an emphasis on performance of the system rather than security when compared to the traditional telephone system.

This research aimed to enhance the security of voicemail on an Asterisk server. We also evaluated the performance of the server based on the response time and the failed call rates when there were a number of concurrent calls as it would indicate the availability of the services. In addition, the Transport Layer Security was chosen for a secure connection between the caller and the Asterisk server, and the voice message was encrypted with the AES over SSL.

According to the standard SPECSIP, we conducted two experiments that measured the performance of the completed call and the voicemail call scenario for both the baseline and the proposed systems. The performance metrics were the call setup delay and the failed call rates, and the number of concurrent calls was varied from 50 to 500. The experimental results showed that the response time of both systems clearly increased when having more concurrent calls but the failed call rates were slightly different. Moreover, for the voicemail calls, the response time had no difference when retrieving voice messages whereas the encrypted voice messages affected the response time significantly.

KEY WORDS : VoIP / SIP / SECURE VOICEMAIL / ASTERISK / PERFORMANCE EVALUATION

78 pages

การประเมินประสิทธิภาพระบบ Voicemail ที่ปลอดภัยบน Asterisk โดยใช้มาตรฐาน SPECSIP

PERFORMANCE EVALUATION OF SECURE VOICEMAIL ON ASTERISK VIA SPECSIP BENCHMARK

เฉลิมภพ อ่องสิทธิ์ 4836584 ITCS/M

วท.ม. (วิทยาการคอมพิวเตอร์)

คณะกรรมการที่ปรึกษาโครงการวิจัย : สุดสงวน งามสุริยโรจน์, Ph.D., วัสกา วิสุทธิวิเศษ, Ph.D., สรณันท์ จิวะสุรัตน์, Ph.D.

บทคัดย่อ

เทคโนโลยีด้าน VoIP ได้รับการพัฒนาและนำมาใช้งานจนเป็นที่ยอมรับอย่างแพร่หลาย Asterisk เป็นหนึ่งใน open source software ที่ได้รับความนิยมมากในการใช้งานด้านระบบโทรศัพท์พร้อมบริการ Voicemail ที่มีความสามารถอย่างหลากหลาย สนับสนุนการทำงานกับโปรโตคอล SIP ซึ่งการมุ่งเน้นพัฒนาด้านประสิทธิภาพการทำงานให้เทียบเท่าระบบโทรศัพท์พื้นฐาน ทำให้ความสำคัญในด้านความปลอดภัยเป็นสิ่งที่สำคัญรองลงมา

งานวิจัยนี้มีจุดมุ่งหมาย ในการปรับเพิ่มประสิทธิภาพด้านความปลอดภัยให้กับระบบ Voicemail บน Asterisk โดยได้ทำการประเมินประสิทธิภาพจาก call setup delay และ failed call rates ขณะที่มีคู่สายที่ใช้บริการพร้อมกันจำนวนมาก ซึ่งบ่งชี้ถึงความสามารถในการให้บริการได้ Transport Layer Security ได้ถูกเลือกใช้สำหรับการเชื่อมต่อแบบปลอดภัยระหว่างผู้ใช้งานและ Asterisk และเพิ่มความปลอดภัยของข้อความเสียง ด้วยการทำ AES encryption ที่รับ-ส่งข้อมูลกันผ่าน SSL

การทดสอบได้มีการอ้างอิงมาตรฐาน SPECSIP โดยการวัดประสิทธิภาพของ Completed call และ Voicemail call เมื่อเปรียบเทียบระหว่างระบบ Asterisk เดิมกับระบบใหม่ที่ได้นำเสนอ โดยค่าที่ใช้เปรียบเทียบคือ call setup delay และ failed call rates ของจำนวนคู่สายที่ใช้งานพร้อมกันตั้งแต่ 50 คู่สายจนถึง 500 คู่สาย ซึ่งผลการทดลองแสดงให้ว่าเวลาตอบสนองของทั้งสองระบบจะเพิ่มขึ้นอย่างเห็นได้ชัดเมื่อมีจำนวนคู่สายเพิ่มขึ้น แต่อัตราการล้มเหลวของการเชื่อมต่อแตกต่างกันเพียงเล็กน้อย นอกจากนี้สำหรับ การใช้บริการ voicemail เวลาการตอบสนองต่อการใช้งานไม่แตกต่างกันเมื่อมีการเรียกฟังข้อความเสียง ขณะที่กระบวนการเข้ารหัสข้อความเสียงมีผลต่อเวลาการตอบสนองอย่างมีนัยสำคัญ

78 หน้า

CONTENTS

CONTENTS (cont.)

CONTENTS (cont.)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (cont.)

CHAPTER I
INTRODUCTION

The IP protocol has been widely recognized as the default protocol for data and information communication in the modern digital world. It provides many facilities for the communication via the Internet. One significant application, particularly Voice over Internet Protocol (VoIP) or Internet telephony, is in fact the convergence of data and voice services over the IP networks. It allows people to communicate from and to anywhere with the minimal cost. VoIP implementations such as IP-PBX (Private Branch Exchange) based on the popular H323 protocol have been used for decades to make calls effective and cost saving possible. However, current VoIP implementations are mostly based on the newly Session initiation Protocol (SIP). In addition, the popular open source PBX software called Asterisk has been developed to become an alternative way to as it provides the lightly cost but effective communication IP-PBX system.

Furthermore, an additional voice service such as a voice mail system has become a valuable ubiquitous feature, and it can be associated with the existing telephone systems. This service can be used to help communicate among users when any call busy or rang with no answer. A user can leave a message to other users, and can be integrated into an e-mail system. Traditionally, the access to the voicemail service is done via a typical telephone, and it is complicated to handle voice messages. Using graphical screen-based as web-based would be easier and give better utilization of resources.

In this work, we propose the design and implementation of a secure service voice mail. Our work focuses on using soft phones to communicate since it can provide a graphically user friendly interface. We also add the security mechanism for privacy and monitoring. In addition, the system provides some extensions as well as the services for recovering voice messages**.** This work aims to improve the security

and privacy of voice mail communication as well as to provide convenience for effective communication.

1.1 Problem Statement

Voicemail systems usually manage voice messages of missed calls and waiting calls. Nonetheless, developing voicemail services is a complex process and has several obstacles as described below.

(1) *Handling storage for keeping large amount of voice messages*

Since voicemail services have become an essential part of any organization, and with many business calls, voice messages have been rapidly increased. Handling the large amount of such voice messages would be a challenge since there must be sufficient storage to adequately support and still provide convenience for users.

(2) *Limited interaction thru the traditional telephone-based interface*

Traditional telephones inconveniently handles and accesses each voicemail message in voice mailboxes. On the contrary, our voice mail services allow users to efficiently leave, retrieve, store and forward voice messages. It also provides an easy-to-use graphical interface for users via the web.

(3) *Concerning security issues*

Voicemail service could be vulnerable in many ways. It requires encryption and feasible mechanism to reduce these risks. However, the processes must be transparent to the users and easy to apply, in addition, no effect to voice quality.

(4) *Extension and recovering of voice mailboxes*

The voicemail service should provide ubiquitous features and support for rapid growth of number of users. It is also necessary to support the reliability, availability, and scalability of the system. As a result, an effective approach is required for data recovery due to accidental or disaster situation as well as the approach for backing up voice messages and extending the size of voice mailboxes.

1.2 Objectives of the Research Project

The objectives of this project are to design and develop a secure voicemail system as well as to facilitate the system for handling voice messages and for utilizing voicemail services. In summary, the objectives are as follows.

(1) Develop an IP-PBX server using an Asterisk server to provide the voice over IP and fully voicemail services.
(2) Enhance the security of the Asterisk and voicemail system.
(3) Evaluate the performance of the Asterisk and voicemail system when adding secure communication and encryption feature.

1.3 Scope of the Research Project

This research project is to create a secure voicemail system that provides a variety of effective and user-friendly functions. In summary, the project's scopes are as follows.

(1) To study the concept of IP-PBX using the SIP protocol as the base.
(2) To install an Asterisk server as the IP-PBX server implementation.
(3) To implement voicemail services on Asterisk, and configure its features that offer services to leave, retrieve, and forward messages.
(4) To enhance the security on voice message secrecy with AES encryption while integrating TLS for secure SIP signaling.
(5) To measure and compare the performance of the typical Asterisk with that of the proposed work.

1.4 Organization of the Research Project

This document consists of five chapters, and their description is given below.

Chapter 1: Introduction gives an overview of problems related to the voicemail system development. Specifically, we describe the

motivation, the problem statement, the objectives and the scope of this work.

Chapter 2: **Background** explains some basic background used in our work. They include VoIP, related protocols, IP-PBX, voicemail, CODEC, and the overview of Asterisk architecture.

Chapter 3: **Literature Review** depicts some literature review related to our work. They include voicemail features, VoIP security issues, authentication, and AES voice encryption.

Chapter 4: **Design and Implementation of voicemail system on Asterisk** presents the design and implementation of our Asterisk and voicemail system. We also give the details of major components and their functions designed for the proposed Asterisk and voicemail system.

Chapter 5: **Experiments and Results** describe the experiments of Asterisk and voicemail system and their performance results. The performance of the system is measured and compared with the typical Asterisk system. The results are also discussed.

Chapter 6: **Conclusions** summarize our contribution and suggest future work.

CHAPTER II
BACKGROUND

This chapter provides some background on VoIP (Voice over Internet Protocol), related protocols, and software tools used for developing a voicemail system. We also briefly describe IP-PBX, SIP, IAX, RTP, SRTP, ZRTP, CODEC, and Asterisk PBX, as well as compare VoIP software.

2.1 Voice over IP

The traditional telephone network called PSTN (public switched telephone network) is normally based on circuit switching networks. On the contrary, voice over Internet Protocol (VoIP) [1] is a protocol for transmitting voice through IP networks, and allows voice communication through regular phones, soft phones or cellular phones from anywhere via Internet service providers while saving lots of cost charges that are normally subject to callers. It is also known as IP telephony, Internet telephony, voice over broadband, broadband telephony, and broadband phone. As a result, PSTN and VoIP have different characteristic as shown in Figure 2.1 and 2.2, respectively.

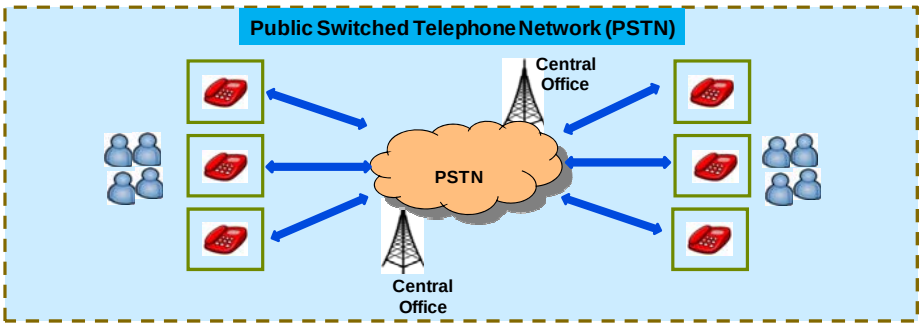


Figure 2.1: PSTN

2.2 IP-PBX

An IP-PBX (Private Branch Exchange) is the software used for the telephone switching system within an enterprise. It provides traditional PBX functionalities to VoIP calls. It can switch calls between a VoIP user and a traditional telephone user, or between two traditional telephone users in the same way that a conventional PBX does.

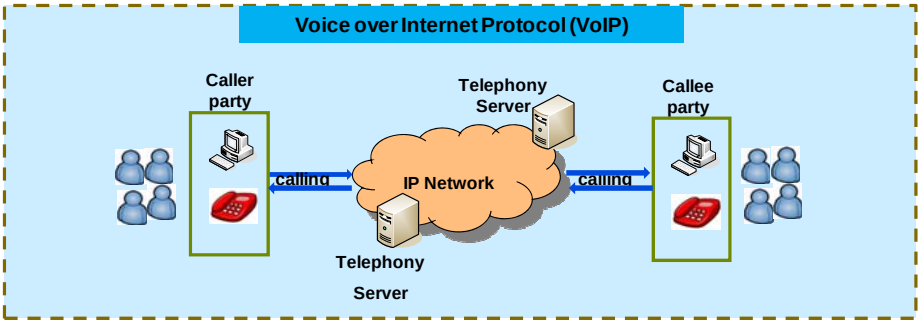


Figure 2.2: VoIP

In addition, it is relatively inexpensive and easy to add extra functionalities such as conferencing, live calls, interactive voice response (IVR), text to speech/automatic speech recognition (TTS/ASR), and PSTN interconnection ability supporting both analogue and digital circuits. VoIP protocols include SIP, Inter-Asterisk eXchange (IAX), H.323, and Jingle (extension of Jabber/XMPP protocol introduced by GoogleTalk).

2.3 Protocols

The mechanism for carrying a VoIP connection generally involves a series of signaling transactions and media streams between the endpoints. Several protocols can be used to implement a VoIP system, and we will discuss some of those that are related to our work below.

2.3.1 Session Initiation Protocol (SIP)

Session Initiation Protocol (SIP) [1] is an application-layer control (signaling) protocol developed by IETF (The Internet Engineering Task Force) for initiating, modifying, and terminating sessions of conversation participants [1]. Sessions include Internet telephone calls, multimedia distribution, and multimedia conferences. The goal of SIP is to provide signaling and call setup protocols for IP networks. It also provides the call processing functions and features in the PSTN. The significant advantage of SIP is a short, simple and flexible protocol, which can be run over both wired and wireless communications.

- SIP Components

Two main components in the SIP architecture are user agents and servers. User agents (UA) are the combination of User Agent Clients (UAC) and User Agent Servers (UAS). An UAC refers to a network element that sends SIP requests and receives SIP responses. Similarly, an UAS is a network element that receives requests in order to service them, and then responds to those requests. It can be classified into four types. First, proxy servers operate on behalf of users and are responsible for routing and delivering messages. Second, redirect servers keep a user database which allows them to inform proxy servers of a user location. Third, location servers are used by a redirect server or a proxy server to obtain information about a caller's possible location. Finally, registrar servers save information about where a party can be found. All server types of SIP components are shown in Figure 2.3.

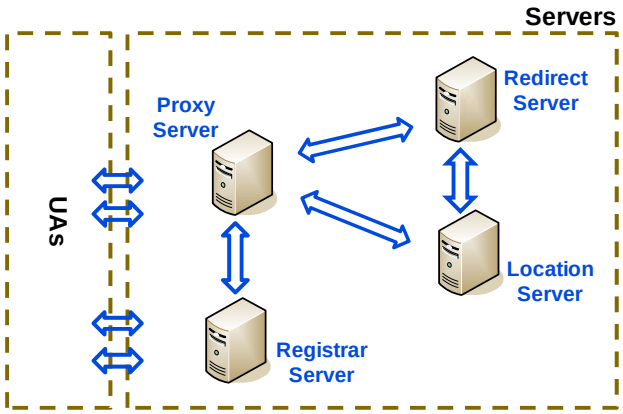


Figure 2.3: Server Types of SIP Component

- SIP Methods

SIP uses methods, requests and corresponding responses to establish a call session. The details are as follows.

(1) SIP requests: There are six basic request methods below:

INVITE: to established a session

ACK: to confirm an INVITE request

BYE: to end a session

CANCEL: to cancel the establishment of a session

REGISTER: to register the user location (host name, IP)

OPTIONS: to give information about the capabilities of the calling and receiving SIP phones

(2) SIP responses to answer SIP requests, and have six classes below.

1xx = informational responses, e.g., 180, which means ringing

2xx = success responses

3xx = redirection responses

4xx = request failures

5xx = server errors

6xx = global failures

- Sample SIP Call

There are four essential steps of a SIP call. They are registration, session set up, media session, and tearing down session. Registration is the first step that each user sends the register message to the SIP server since it will be used for authentication when calling. Next, a session is set up to establish the connection using the INVITE message. If the establishment is successful, the media session allows the end-to-end connection for a call. After the end of a conversation, the connection is ended by a tearing down session. These steps of a SIP call are shown in Figure 2.4.

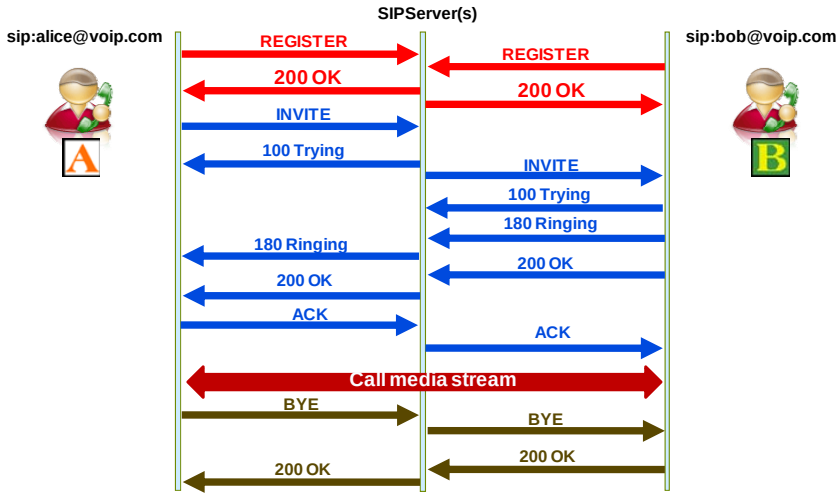


Figure 2.4: Simple SIP call

2.3.2 Inter-Asterisk eXchange Protocol (IAX)

Inter-Asterisk eXchange (IAX) was designed for Asterisk PBX and supported by a number of soft switches and PBX. However, this protocol has not yet been accepted as a standard. It is used to enable VoIP connections between servers as well as client-server communication [23]. IAX now becomes IAX2, and it is an alternative protocol like other VoIP protocols such as SIP and H323. However, the IAX was developed for communicating between Asterisk servers. It is a transport protocol that uses a single UDP port (4569) for both controlling and transmitting streaming media over IP networks. The use of a single UDP port for both signaling and transmitting of media helps keep the minimal number of ports required to open at the firewall. These factors also help make IAX one of the easiest protocols to implement in secure networks. In addition, due to the main design goals, IAX is likely to work behind NAT with lower overhead, lower latency, lower bandwidth required and reduced processing power. Moreover, the IAX can do the authentication in three mechanisms: plain text, MD5 hashing, and RSA key exchange. In the future, IAX may be able to encrypt the stream between endpoints with the use of an exchanged RSA key, or the dynamic key exchange at call setup, and allowing the use of automatic key rollover [23]. This would be very attractive for creating a secure link. Figure 2.5 shows the IAX messages.

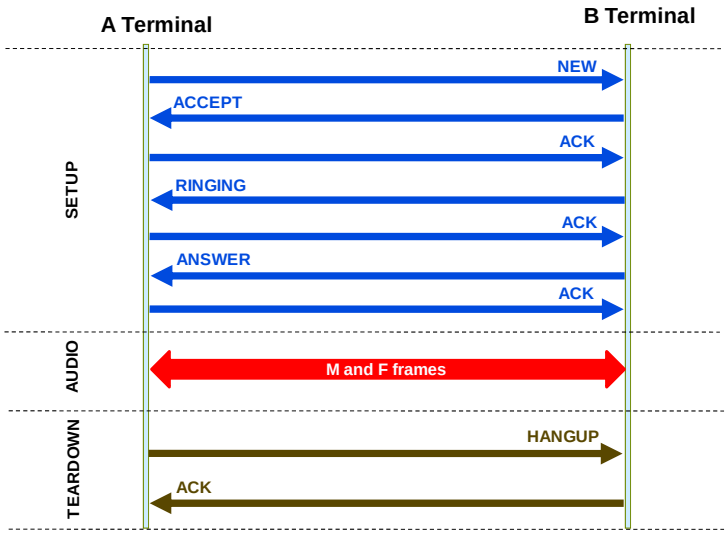


Figure 2.5: IAX Messages

2.3.3 Real-time Transport Protocol (RTP)

RTP [9] is a transport protocol for real-time applications that provides the end-to-end network transport functions suitable for applications transmitting real-time data such as audio, video or simulation data over multicast or unicast network services. RTP does not address resource reservation nor guarantee the quality-of-service for real-time services. The data transport is augmented by a control protocol called RTCP that allows monitoring of the data delivery in a manner scalable to large multicast networks, and also provides the minimal control and identification functionality. RTP and RTCP protocols are designed to be independent of the underlying transport and network layers. They support the use of RTP-level translators and mixers.

2.3.4 Secure Real-time Transport Protocol (SRTP)

Secure Real-time Transport Protocol (SRTP) [10] was recently standardized by IETF as a profile of the RTP protocol since it can provide confidentiality, message authentication, and replay protection to the RTP traffic and to the RTCP protocol.

SRTP provides a framework for encryption and message authentication of RTP and RTCP streams. SRTP defines a set of default cryptographic transforms, and allows new transforms to be introduced in the future. With appropriate key

management, SRTP is secure for unicast and multicast RTP applications. SRTP can also achieve high throughput and low packet expansion. It proves to be a suitable protection for heterogeneous environments of mixed wired and wireless networks. To get such features, default transforms are described, based on an additive stream cipher for encryption, a keyed-hash based function for message authentication, and an "implicit" index for sequencing/synchronization of RTP sequence number for SRTP and an index number for secure RTCP (SRTCP).

The security goals for SRTP are to ensure the confidentiality of the RTP and RTCP payload, and the integrity of the entire RTP and RTCP packets as well as the protection against replayed packets. Particularly, SRTP provides message confidentiality, authentication, integrity checking and replay protection for RTP traffic on a per-packet basis. The standard specifies AES encryption of the RTP payload, the message authentication hash of the header and the encrypted payload using HMAC or SHA1 [10]. The HMAC is truncated to the 32-bit authentication tag and appended to the packet before transmission. The receiver repeats the HMAC computation on the received header payload and compares it to the tag. A positive match not only confirms data integrity but also authenticates the sender.

2.3.5 ZRTP

ZRTP [11] is a protocol for media path Diffie-Hellman exchange to agree on a session key and parameters for establishing SRTP sessions. The ZRTP protocol is media path keying because it is multiplexed on the same port as RTP and does not require support in the signaling protocol. In addition, ZRTP does not assume a Public Key Infrastructure (PKI) or require the complexity of certificates at the end devices. For the media session, ZRTP provides the confidentiality and the protection against man-in-the-middle attacks in cases where the signaling protocol provides the end-to-end integrity protection and authentication. Moreover, ZRTP can utilize a Session Description Protocol (SDP) attribute to provide discovery and authentication through the signaling channel. To provide the best effort SRTP, ZRTP utilizes normal RTP/AVP profiles.

2.4 CODEC

A codec [25] is a device or program capable of encoding and decoding a digital data stream or signal. This term may be a combination of compressor-decompressor, coder-decoder or compression-decompression algorithm. In the mid 20th century, a "codec" was a hardware device that coded analog signals into pulse-code modulation (PCM) and decoded them back. Recently, it has been developed as software, and CODEC refers to a technique to encode or decode a data stream or signal for transmission, storage or encryption. It is often used in videoconferencing and streaming media applications. There are several CODEC standards and their features as shown in Table 2.1 below.

Table 2.1: CODEC Standards

CODEC	Bandwidth (B/W)(kbps)	Packet Interval (ms)	Ethernet Overhead B/W (kbps)	Processing Intensity	Total Bandwidth (kbps)
G.711	64	20	31.2	Low	95.2
G.726	32	20	31.2	Medium	63.2
G.728	16	10	31.2	High	78.4
G.729A	8	10	31.2	High	39.2
GSM	13	20	31.2	Medium	44.2
iLBC	15	10	31.2	High	46.2
Speex	8-32	10	31.2	High	39.2

2.5 Voicemail

Voicemail [22] is an electronic telephone messaging service. It allows message leaving for non-simultaneous or busy calls between two or more individuals. This significant service has many potential advantages including the fulfillment of the effective communication system. Traditionally, voicemail systems are directly connected to PBXs via proprietary protocols and can be accessed only by means of legacy phones and DTMF signaling. Currently, most systems prefer to handle voice mail messages via an e-mail that can provide the graphical user interface. However,

the strong security mechanisms are required when it is transformed to the computerized work. The well-known voicemail software is found in Skype, Google VoiceMail on Gmail, Google Talk and many others.

2.6 Asterisk

Asterisk [21] is a complete software PBX developed by Mark Spencer of Digium, Inc in 1999. Originally, it was designed for Linux but now runs on a variety of operating systems including NetBSD, OpenBSD, FreeBSD, MacOS, and Solaris. Asterisk needs no additional hardware for VoIP although it does expect a non-standard driver that implements dummy hardware as a non-portable timing mechanism for certain applications such as conferencing. Asterisk also supports a wide range of video and VoIP protocols including SIP, MGCP and H.323. Asterisk also has its own protocol called Inter-Asterisk eXchange (IAX2) for efficient trunks of calls among Asterisk servers, and for other VoIP service providers who support it. It provides many features available in proprietary PBX systems including voice mails, conference calls, interactive voice responses or phone menus, and automatic call distribution.

In addition, users can create new functionalities by writing dial plan scripts in Asterisk's own extensions languages, by adding customized loadable modules written in C, or by implementing Asterisk Gateway Interface (AGI) programs using any programming language capable of communicating via the standard streams system, or by using network TCP sockets.

2.6.1 Asterisk Architecture

Asterisk architecture consists of two components: APIs and a central PBX core system as shown in Figure 2.6. Since Asterisk is carefully designed for maximum flexibility, specific APIs are defined around a central PBX core system. This advanced core handles the internal interconnection of the PBX, cleanly abstracted from specific protocols, codecs, and hardware interfaces from the telephony applications. This allows Asterisk to use any suitable hardware and technology available at present or in the future to perform its essential functions, connecting hardware and applications. Each component is described in turn below.

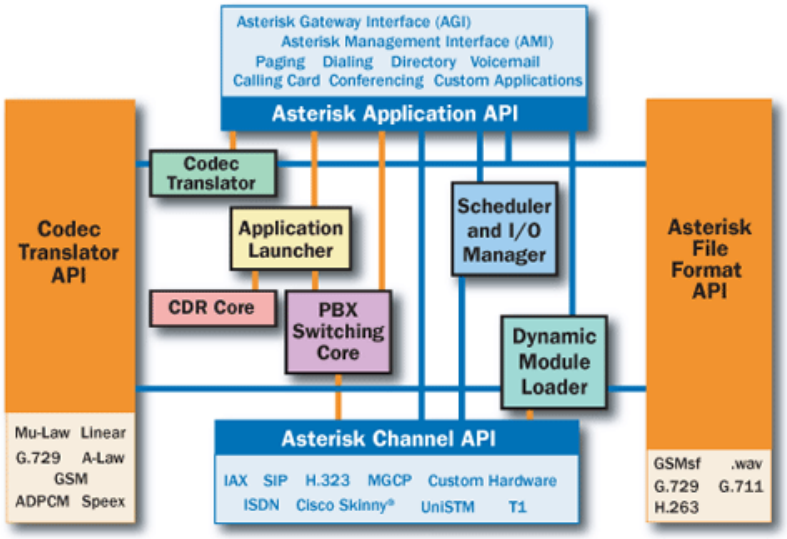


Figure 2.6: Asterisk Architecture

- **Central PBX core system**

  (1) PBX Switching - The essence of Asterisk is a Private Branch Exchange switching system that is used to connect calls between various users and automated tasks. The switching core transparently connects callers arriving on various hardware and software interfaces.

  (2) Application Launcher – Applications are launched to perform services for users including voicemails, file playback, and directory listing.

  (3) Codec Translator - Codec modules are used for encoding and decoding of various audio compression formats used in the telephony industry. A number of codecs are available to suite diverse needs and have the best balance between audio quality and bandwidth usage.

  (4) Scheduler and I/O Manager – This is used for handling low-level task scheduling and system management for optimal performance under all load conditions.

(5) Loadable Module APIs - Four APIs are defined for loadable modules, facilitating hardware and protocol abstraction. Using loadable modules, the Asterisk core does not have to worry about the details of how a caller is connecting or what codecs are in use, etc.

- **APIs**

(1) Channel API handles the type of connection that a caller is arriving on; either it is a VoIP connection, ISDN, PRI, Robbed bit signaling, or some other technology. Dynamic modules are loaded to handle the lower layer details of these connections.

(2) Application API allows various task modules to perform various functions including conferencing, paging, directory listing. voicemails, in-line data transmission, and any other tasks which a PBX system might currently perform or in the future as they are handled by separate modules.

(3) Codec Translator API are loaded to support various audio encoding and decoding formats such as GSM, Mu-Law, A-law, and even MP3.

(4) File Format API handles the reading and writing various file formats for the storage of data in the file system.

Using these APIs, Asterisk achieves a complete abstraction between its core functions as a PBX server system and various existing technologies in the telephony arena. The modular form allows Asterisk to seamlessly integrate both currently implemented telephony switching hardware and the growing packet voice technologies emerging today. In addition, the ability to load codec modules allows Asterisk to support both the extremely compact codecs necessary for packet voice over slow connections such as a telephone modem, and still be able to provide high audio quality over less constricted connections.

The application API provides flexible use of modules to perform any function on demand, and opens a venue for new application development to suite unique needs and situations. In addition, loading all application modules allows the system flexibility, and allows the administrator to design the best suited path for callers on the PBX system and modify call paths to suite the changing communication needs.

2.6.2 Asterisk Configuration and Dial Plan

The Asterisk configuration is organized to control the behavior of all connections through the PBX. It is generally handled using *.conf files. In addition, it can be handled from a database (MySQL or PostgreSQL) or templates files. For example, the configuration file “extensions.conf” containing the dial plan of Asterisk is used to control the execution flow for all operations, whereas the configuration file “sip.conf” contains the setting of SIP protocol.

A dial plan defines a set of dialing rules which the instruction format can be written and described in detail below.

exten => Extensions, Priorities, Applications(,arguments)

All rules begin with the reserved word “exten =>” and having a set of key attributes. *Extensions* refer as a phone number, and *Priorities* refer as a sequence of instructions to be run step by step, and *Applications* refer as the execution using the assigned *arguments*. Figure 2.7 depicts an example of a dial plan. The first line shows that, when someone calls the phone number “1001”, the call acts to dial for 40 seconds. If the call has no answer, the next instruction is executed; that is, the voicemail is run to leave a voice message to the assigned voicemail box.

extensions.conf

```
[default]
exten => 1001,1,Dial(SIP,1001,40,r)
exten => 1001,2,VoiceMail(b1001@default)

exten => 1002,1,Dial(SIP,1002,40,r)
exten => 1002,2,VoiceMail(b1002@default)

exten => 8500,1,VoiceMailMain()
```

Figure 2.7: A Dial Plan

2.7 Comparison of VoIP Software

VoIP software is used to conduct telephone-like voice conversations across IP networks. It can be classified by the type of software as general soft phone clients, mobile phones, frameworks and libraries and server software. In addition, each type of software is compared based on OS support, license, protocols, encryption, and other capabilities such as target markets and latest release. The comparison of products of soft phone clients and server software is shown in Table 2.2 and 2.3, respectively, below.

Table 2.2: Comparison of Soft Phone Clients

Program	OS	License	Protocols/ Based upon/ compatible with	encryption	Other capabilities
CounterPath X-Lite	Windows,Mac, Linux	Freeware/ Closed proprietary	SIP, STUN, ICE	Unknow	IM, single login account, for Windows and Mac also conferencing, Viedo and SIMPLE based presence
Minisip	Windows XP, 2000, Linux, Windows Mobile	GPL/ LGPL Free software	SIP	SRTP,TLS, MIKEY(DH,PSK,PKE), end-to-end encryption	Video, IM conferencing
Skype	Windows XP, 2000, Linux, Windows Mobile	Freeware/ closed proprietary	Proprietary P2P protocol	yes	Video, file transfer, voicemail, Skype to phone, phone to Skype
Twinkle	Linux	GPL free software	SIP	SRTP, ZRTP	
Zfone	Linux, Mac OS X, Windows	Viewable source/ Proprietary license	SIP,RTP	SRTP, ZRTP	

Table 2.3: Comparison of Server Software

Program	OS	License	Protocols/ Based upon/ compatible with	encryption	Other capabilities	Key and target markets
Asterisk PBX	Linux, BSD, Mac OS X, Solaris	GPL free software / proprietary	SIP, H.323, IAX, MGCP		IVR, call conferencing, call queuing, voicemail with directory	
FreeSWITCH PBX Sortphone	BSD,Linux, Mac OS X, Solaris,Windows	Open source, Mozilla Public License, free software	SIP,STUN,XMPP, jabber, IAX, H.323, Skype	TLS, SRTP, ZRTP	Recording, Voicemail, Jabber IM, Google Talk, Proxy, Media Gateway, Soft-PBX, IVR	Large soft-switch users, home PBX users, softphone users
SIP Express Router (SER)	BSD,Linux,Solaris	GPL free software	SIP		SIP Registrar/Proxy, Authentication, Diameter RADIUS, ENUM	SIP Service Providers
Kamailio/OpenSIPS (OpenSER)	BSD,Linux,Solaris	GPL free software	SIP, Jabber, XMPP		•SIP Registrar/Proxy, Authentication, Diameter RADIUS, ENUM, least-cost-routing	SIP Service Providers

CHAPTER III
LITERATURE REVIEW

This chapter provides the literature review related with our work. They include security issues in voice mails [3], AES voice encryption [4], user authentication in SIP [5], SIP authentication scheme using ECDH [6], the implementation and evaluation of SIP-based secure VoIP communication system [7] and a practical analysis of Asterisk SIP server performance [8].

3.1 Security Patterns for Voice over IP Networks

The paper [3] presents the architecture and use cases in the VoIP infrastructure. They describe the security patterns with mechanisms that can control many possible attacks and could be used to design a secure system. There are four mechanisms that work together. They are network segmentation, VoIP tunneling, signed authenticated calls, and secure VoIP calls. Figure 3.1 shows the relationship between such mechanisms and related cryptographic patterns as indicated with double lines. The paper focuses on two significant security features: message secrecy and authentication.

Particularly, the secure VoIP call mechanism hides the transmitted messages by encrypting calls in the VoIP environment. The class diagram of Figure 3.2 shows the secure channel communication in VoIP. This model uses the flow to indicate the choice of encryption algorithms. Both the caller and the callee use the same set of algorithms although they are shown only on the caller side.

The symmetric algorithms such as DES, 3DES and AES are preferred for data encryption using a shared secret key. However, the Secure Real-Time Protocol (SRTP) and IPSec can be used for encrypting media traffic where the Multimedia Internet KEYing (MIKEY) can be used for exchanging keying materials in VoIP.

3.2 AES Voice Encryption

This paper [4] presents the use of AES algorithm to encrypt voice signal while it does not manage the voice transmission. Thus, key exchange protocols were not addressed. Table 3.1 shows the requirements and solutions presented.

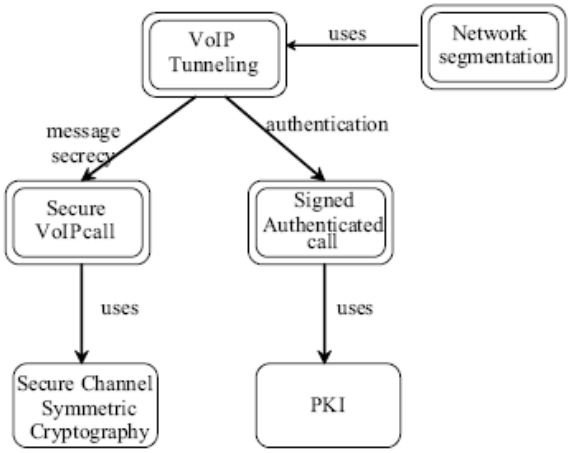


Figure 3.1: Relationships between VoIP Security Patterns [3]

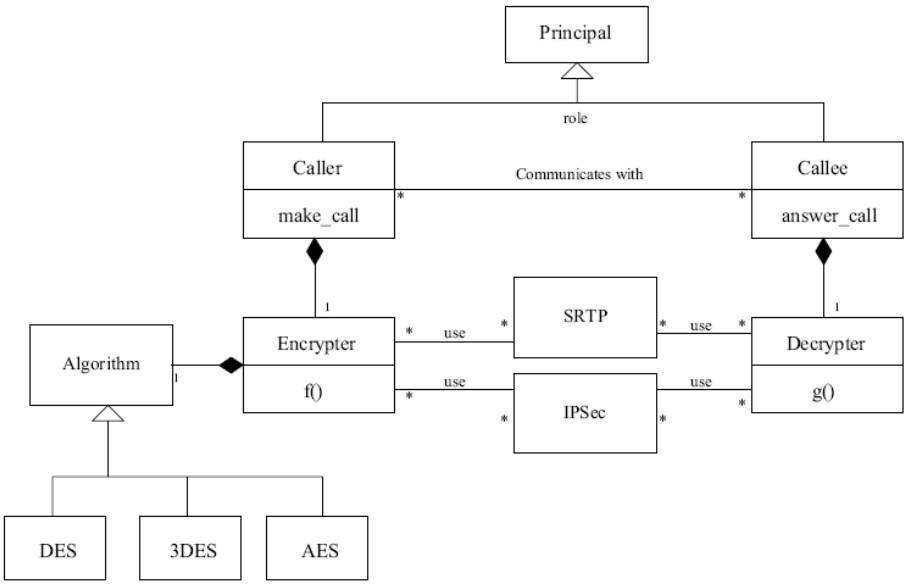


Figure 3.2: Class Diagram for a VoIP Secure Channel [3]

Table 3.1: Requirements and Solutions for Voice Encryption

Requirements	***Solution***
• **AES** • **Voice Encryption** • **128-bit key**	• Input Voice into PC • Save File • Sample Voice

• **8 thousand samples/second (decent quality sound)** • **Obtain original signal via Decryption**	• Encrypt using AES • Save File • Decrypt using AES • Save File • Rewrite into sound file

The main process can be implemented as shown in Figure 3.3. Two functions: real-time encrypted transmission and wireless transmission can be further improved.

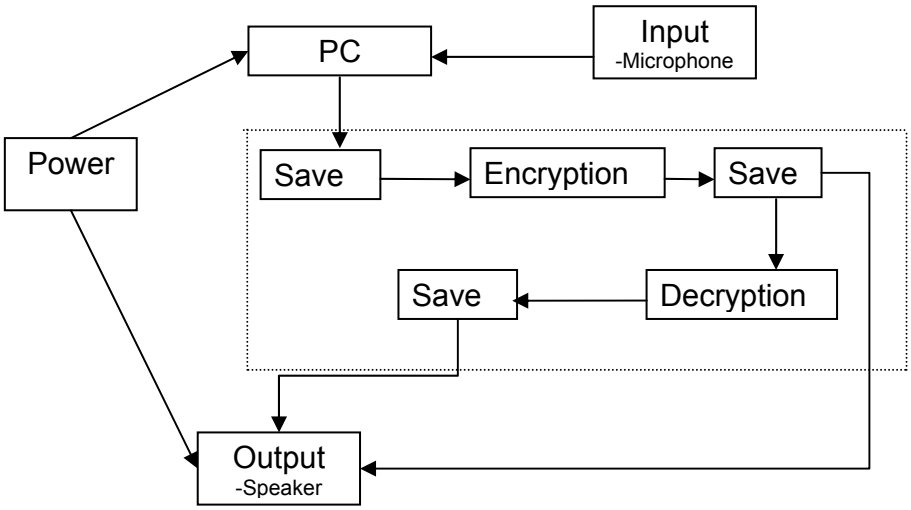


Figure 3.3: AES Voice Encryption

3.3 User Authentication in SIP

The paper [5] examines the current solutions in SIP authentication as there are several existing mechanisms. It also analyses their advantages and disadvantages. However, the paper emphasizes on the HTTP digest based authentication which is also called the SIP authentication.

HTTP digest is generally a challenge-response protocol in a way that a challenge generates a nonce value to the target and a response returns a checksum of the username, password, the nonce value, HTTP method and the requested URI. SIP applies this mechanism for authenticating users to users or users to proxies. A sample flow of SIP authentication mechanism is shown in Figure 3.4.

However, there are some problems of the HTTP digest authentication in SIP, and two major weaknesses are presented. They are the lack of securing all headers and parameters in SIP that needs protection, and the requirement of having pre-existing user configuration on servers which does not scale well.

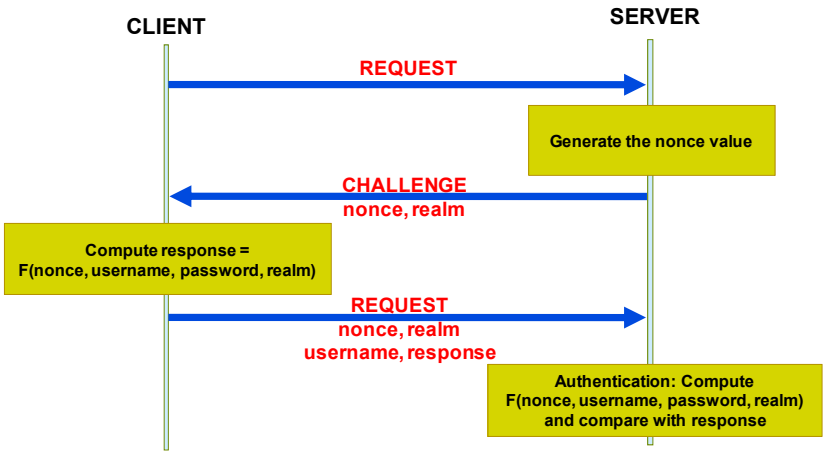


Figure 3.4: SIP Authentication Mechanism based on HTTP Digest [5]

3.4 SIP Authentication Scheme using ECDH

The paper in [6] presents a new approach for secure SIP authentication using a public key mechanism called Elliptic Curve Cryptography (ECC) since ECC provides smaller key sizes and faster calculations than other public key infrastructures at the same security level. The proposed approach also combines the security and the quality-of-services issues.

In general, SIP authentication is a lightweight registration process relying on a challenge/response mechanism that employs a pre-shared password. The HTTP digest authentication depends on HTTP features that username and passwords are clearly transmitted as plaintext. Thus, the scheme is weak due to the off-line password guessing attacks, server spoofing and replay attacks.

ECDH approach for secure SIP authentication is proposed as shown in Figure 3.5 where SIP client and SIP server has a pre-shared password for authorization. Both the server and the client have an elliptic curve public key pair

where “d” represents the private key and “Q” represents the public key. Other global domain parameters for both parties are as follows. “G” is the base point in the curve, a and b are coefficients, “q” is the field, “h” is the cofactor, and “n” is the order of G for the selected curve. With the coefficients, a and b, the selected elliptic curve is in the form of Equation (1) below.

$$y^2 = x^3 + ax + b \qquad (1)$$

Equation (1) is determined and chosen according to the NIST’s recommended elliptic curves which are robust against those attacks.

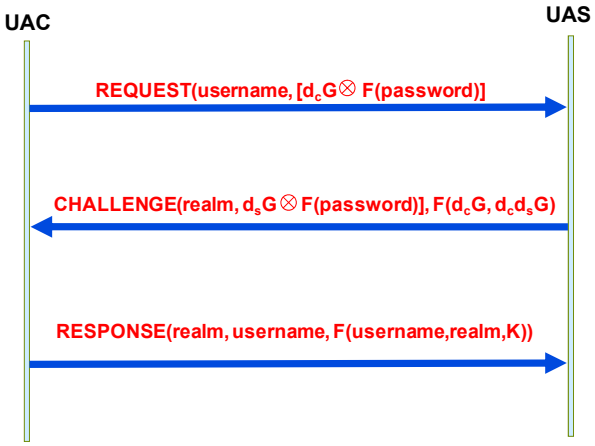


Figure 3.5: ECDH approach [6]

At the first step, the SIP client sends a REQUEST message including its username and public key xor’ed with its hashed password. After the SIP server receives the REQUEST message, the following operation in Equation (2) is performed to compute the d_cG back.

$$d_cG = [(d_cG \otimes F(password) \otimes F(password))] \qquad (2)$$

Next, the SIP server creates a session key, K, according to Equation (3) using its private elliptic key and the result of Equation (1), and sends a CHALLENGE message to the SIP client including realm, its public key xor’ed with the pre-shared hashed password, the hashed pair of the client’s public key and the session key, K.

$$K = (d_s.d_cG) \quad (3)$$

At the third step, the SIP client calculates the value of d_sG according to Equation (4). The session key, K, is also calculated by the SIP client using its private elliptic key with the result of Equation (4). In this way, the SIP client can validate the authentication of the SIP server.

$$d_sG = [(d_sG \otimes F(password) \otimes F(password))] \quad (4)$$

After that, the SIP client sends a RESPONSE message consisting of the username, the realm and the hashed triple of username, realm and K to the server. The SIP server can easily verify the RESPONSE message by comparing the username, realm, and K hash value by its own calculation.

The comparison of DH and ECDH shows that ECDH is faster than DH. It is also more efficient in embedded devices like smart cards, wireless devices or PDA due to the memory usage and fast calculation. It convinces that the suggested authentication scheme using Elliptic Curve derived from the Diffie-Hellman key exchange offers the same robust structure against the offline password guessing and server spoofing attacks. It is also efficient and preferable in the applications or devices that require low memory and fast transaction.

3.5 Implementation and Evaluation of SIP-based Secure VoIP Communication System

The paper in [7] proposes and implements a secure VoIP communication system. It also evaluates the performance of the system in two aspects: call setup delay and voice quality. The proposed system requires user authentication, confidentiality and integrity of signaling messages or media stream. Four security categories for SIP based VoIP are SIP signaling security, media security, key management, and security mechanism negotiation for SIP signaling, and they are described briefly below.

(1) SIP signaling security

- Transport Layer Security (TLS) is a protocol used for user authentication and encrypted communication between clients and servers. It provides the hop-by-hop security such as UA-to-SIP proxy or SIP proxy-to-SIP proxy.
- Secure/Multipurpose Internet Mail Extension (S/MIME) provides the end-to-end security between UAs and three security methods: SIP message encryption, electronic signature, and encryption with signature.

(2) Media security

Secure Real-time Transport Protocol (SRTP) is intended to provide encryption, message authentication and integrity, and replay protection to the RTP data. SRTP can achieve high throughput and low packet expansion. It is also proven to be suitable protection in heterogeneous environment.

(3) Key management for SRTP

A security protocol for protecting real-time applications running over RTP needs a key management solution to exchange keys and related security parameters. In addition, Multimedia Internet KEYing (MIKEY) provides three key exchange methods: Pre-shared key, Public-key encryption, and Diffie-Hellman.

(4) Security Mechanism Negotiation for SIP signaling

Security protocols include facilities to agree on the mechanisms, algorithms, and other security parameters. RFC 3329 for Security Mechanism Agreement for the SIP defines the security mechanism negotiation for SIP signaling protection. It is intended to work only between a UA and its first-hop SIP entity (SIP proxy or UA). The OPTIONS method shown as the 494 response message in Figure 3.6 is used for negotiation.

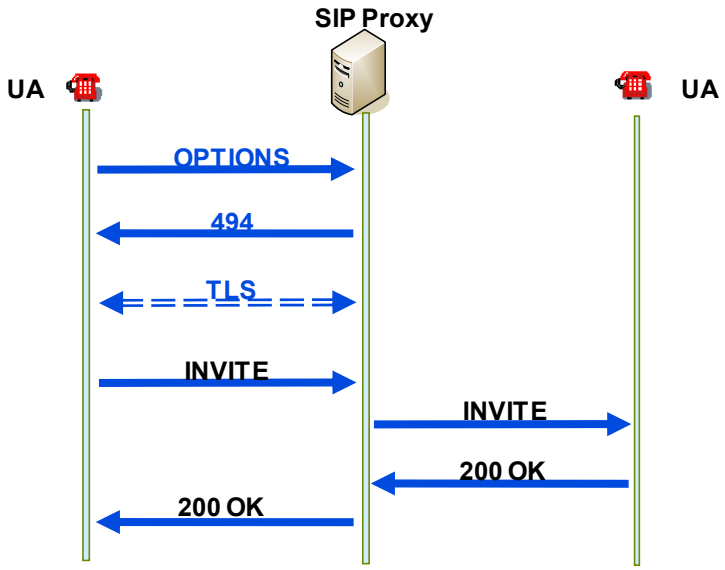


Figure 3.6: Negotiation Process of Security Mechanism [7]

The implementation of the proposed secure VoIP communication system is shown in Figure 3.7 while its performance metrics measured are call set up delay and voice quality as explained below.

- **Call setup delay** is defined as the time between a call request and the first media message transport. It compares the time for a normal call setup and the time for a secure call setup. A secure call setup is defined as the security mechanism negotiation and the call set up with a key exchange among secure VoIP communication flow. The security protocols assigned for SIP signaling message are TLS, S/MIME, SRTP, and MIKEY for media security as shown in Figure 3.8, and the performance results are shown in Table 3.2.

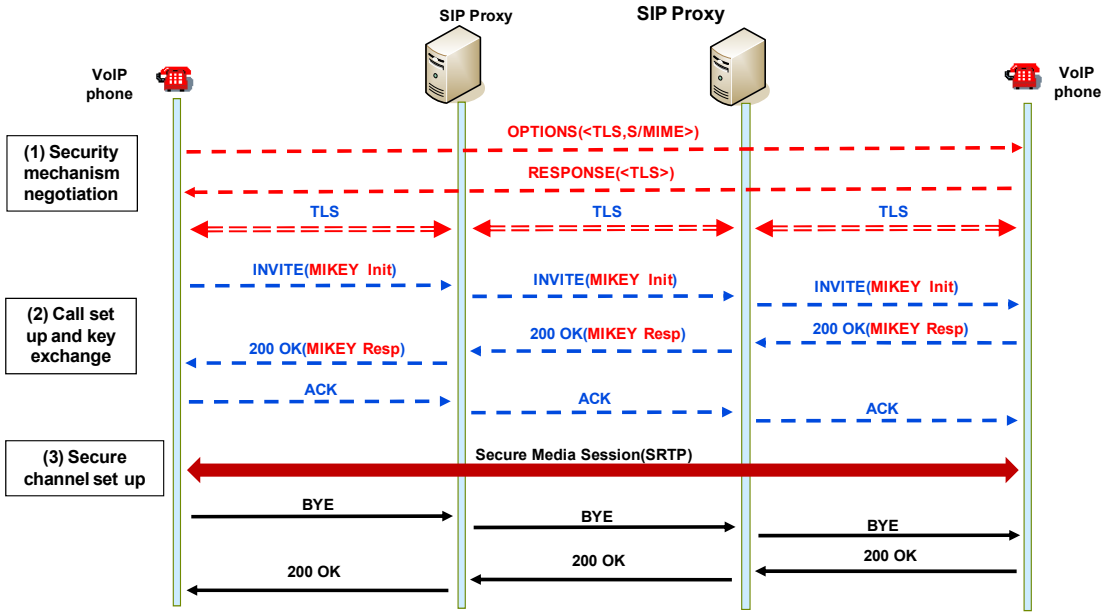


Figure 3.7: Secure VoIP Communication Flow [7]

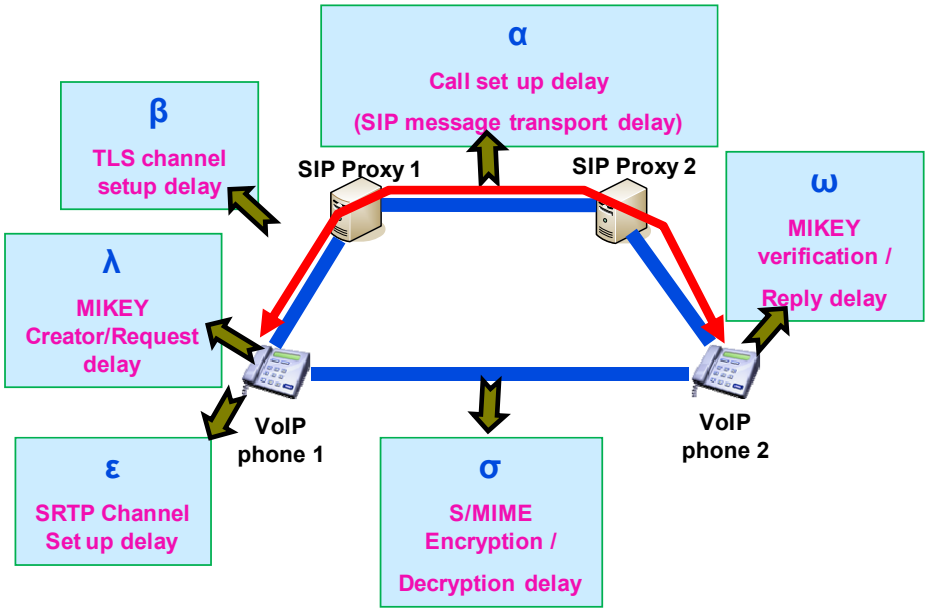


Figure 3.8: Factors for Call Setup Delay [7]

Table 3.2: Performance Results of Call Setup Delay [7]

Security protocol		Require time for call setup	Average time (msec)
signaling	media		
UDP	RTP	α	121.6
UDP	MIKEY(PSK)+SRTP	α + λ + ω +ε	493.875
UDP	MIKEY(PKE)+SRTP	α + λ + ω +ε	544.789
UDP	MIKEY(DH)+SRTP	α + λ + ω +ε	840.604
TLS	MIKEY(PSK)+SRTP	α + β + λ + ω +ε	696.41
TLS	MIKEY(PKE)+SRTP	α + β + λ + ω +ε	742.26
TLS	MIKEY(DH)+SRTP	α + β + λ + ω +ε	1043.38
S/MIME	MIKEY(PSK)+SRTP	α + δ + λ + ω +ε	591.23
S/MIME	MIKEY(PKE)+SRTP	α + δ + λ + ω +ε	642.837
S/MIME	MIKEY(DH)+SRTP	α + δ + λ + ω +ε	938.486

- **Voice quality** is defined as the time required for processing voice packets in VoIP phone (so called voice processing delay) and R-value, MOS for voice communication. It investigates how much the applied security protocols affect the voice quality. For measurement, it identifies the values at the VoIP QoS tester point during the voice exchange (RTP or SRTP) as shown in Figure 3.9.

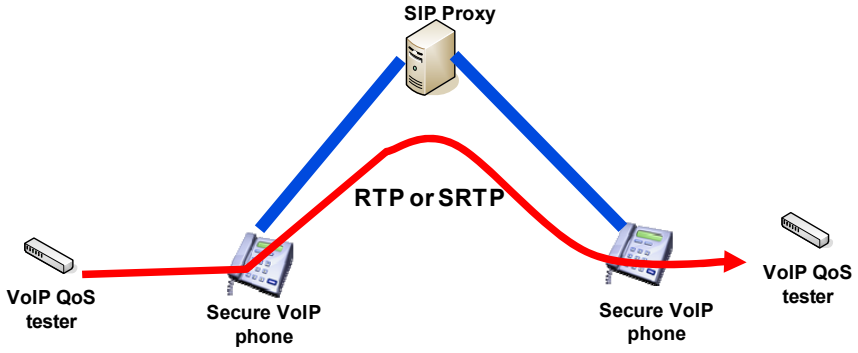


Figure 3.9: Environment for Voice Quality Measurement [7]

The results shown in Table 3.3 illustrate that the voice processing delay for secure VoIP communication increases 18% at the point of the initiator's phone, and does 79% at the point of the responder's phone. It also analyzes the R-value and MOS in both normal and secure VoIP communication. However, it concludes that the effect of the security protocol to the voice quality is insignificant.

Table 3.3: Results of Voice Quality Measurement [7]

Voice quality		RTP	SRTP
Voice processing delay	Initiator's phone (msec)	1.1	1.3
	Responder's phone (msec)	1.4	2.5
R-value		91.5	90.6
MOS(Mean Opinion Score)		4.24	4.22

3.6 A Practical Analysis of Asterisk SIP Server Performance [8]

This paper [8] presents the analysis of the Asterisk server performance as a SIP server. It evaluates the performance of bandwidth consumption and processor load under several utilization scenarios. The bandwidth requirements and the processor load increase significantly since they mean for providing confidentiality, integrity, and availability. A certain type of codec as the voice quality closed to the classic telephony is required to test the server load. The minimum VoIP requirement for the Asterisk SIP server infers that the maximum number of possible calls and the maximum bandwidth can be analyzed.

The experimental setup for performance evaluation consists of the hardware and software configurations as shown in Table 3.4. The details of the Asterisk configuration are shown in Table 3.5. The measurement is done with some constraints explained below.

- **General bandwidth constraints**

ITU-T G.711 is the most frequently used codec in present-day telephony system. It digitally transmits voice that uses 64 Kbps of the available bandwidth. In real usage, there is an extra overhead in addition to the voice traffic, and it leads to the

real bandwidth usage of 87.2 Kbps. In fact, the bandwidth in both directions requires the minimum 128 Kbps for this codec.

Table 3.4 Hardware and Software Configuration

Hardware Configuration		**Software Configuration**	
2 PCs CPU	*Pentium(R) D CPU 2.80GHz*	OS(Server)	*Debian 4.0 kernel 2.6.18-6-686*
Memory	*1 GB RAM*	OS(Client)	*Windows XP[v5.1.2600]*
Network	*NICs 1Gbps/ Switch 3560g*	SIP Server	*Asterisk PBX v1.4.18*
		Test tool	*SIPp v3.1*
		Softphone	*X-lite v3.0*

Table 3.5 Asterisk Configuration

extensions.conf	**sip.conf**

[default]
 	
exten => 2005,1,Ringing; //ring
exten => 2005,2,Wait(10); //wait 10 sec
exten => 2005,3,Playback(vm-nobodyavail); //after 10 sec IVR
exten => 2005,4,Playback(vm-goodbye);
exten => 2005,5,Hangup;
 	
exten => 2006,1,Dial(SIP/user1,10);
exten => 2006,2,Playback(vm-nobodyavail);
exten => 2006,3,Playback(vm-goodbye);
exten => 2006,4,Hangup;
 	
exten => 2007,1,Dial(SIP/user2,10);
exten => 2007,2,Playback(vm-nobodyavail);
exten => 2007,3,Playback(vm-goodbye);
exten => 2007,4,Hangup;
	[user1]
	username = user1
	secret = 1234
	callerid = user1
	type = friend
	context = default
	host = dynamic
	disallow = all
	allow = alaw
	
	[user2]
	username = user2
	secret = 1234
	callerid = user2
	type = friend
	context = default
	host = dynamic
	disallow = all
	allow = alaw

- **External connection bandwidth constraints for the maximum call rate**

The experiment configuration may support up to 250 simultaneous calls which are equivalent to 500 simultaneous channels. Thus, the bandwidth required becomes 43,600 Kbps (as 87.2 Kbps for each channel). If the T3 connection (44,736 Kbps) is used, about 256 calls can be simultaneously initiated, and the availability of

513 channels is given. Therefore, one Asterisk server is sufficient to saturate such a connection.

The processor load has the significant effect since the load is lower for calls initiated by the server and it is increased during the IVR activation period for the voice transmission. It indicates that the overall traffic is 16 Mbps for SIP packets and the overall traffic of about 8 Mbps for real voice traffic (RTP).

The evaluation has the following performance metrics and the results as discussed below.

- **Performance for lower call rates**

The system load does not increase significantly for the higher rate of calls per second either 50 or 100 calls per second. For 50 calls per second, the processor load reaches the maximum of 5%, and for 100 calls per second, the processor load reaches up to 7%.

- **Performance for excessive call rates**

If the limit of 250 calls per second is exceeded, such as attempting to initiate 300 calls per second, certain SIP requests cannot be supported and this leads to retransmissions and some calls are lost. For the call rates of 300 calls per second and above, the retransmissions are initiated after 12 seconds.

The non-linear relationship is found between the call rate and the processor load for the Asterisk server. Hence, it is possible that security policies requiring significant resources can be accommodated by the relatively slight decline in the call rates in order to avoid the maximum levels.

CHAPTER IV
PROPOSED WORK

In this chapter, we propose the design and implementation of a secure voicemail system on an Asterisk server which is our IP-PBX server. The goal is to enhance the security for privacy and monitoring of Asterisk and voicemail services. The details of the proposed work are described as follows.

4.1 Proposed Work

The proposed work aims to handle two issues: system features and security improvement. The system features include the Internet telephony with fully voicemail services and GUI supported. The system can also conveniently access via a traditional telephone, a soft phone and a web browser. The voicemail services store and handle voice messages in a database. The security improvement specifically handles four issues. They are PBX configuration, SIP messages, the media path, and voicemail messages. The details are briefly described as follows.

Secure configuration refers to keeping the value in the configuration files into a database. The plain text password on each configuration file will be encrypted with MD5. Voice messages are encrypted using the AES algorithm. It also includes the security protocol for transporting the protection of SIP messages on the secure channel using TLS. Eventually, the encrypted voice messages are stored into a database and will be decrypted to a specific voice mailbox.

The voicemail services store and handle voice messages in a database. The encrypted voice message is put into the database after users leave a message. The voicemail messages are subsequently decrypted during the retrieved time.

4.2 Physical Overview

The physical overview of the proposed work consists of two main layers as shown in Figure 4.1. The client layer is part of an external connection. A user can access to the system via a web browser, a soft phone, and a telephone. The server layer consists of a web server, an Asterisk server, and a voicemail server. They are briefly described below.

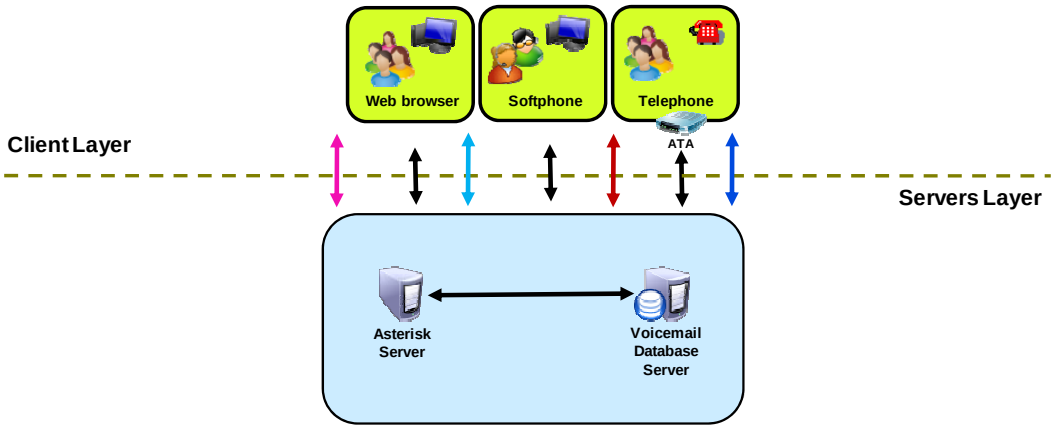


Figure 4.1: Physical Overview

- Web Server provides the web-based access for administration and visual voicemail services such as e-mail.
- Asterisk server provides telephony services including voicemail features for user groups.
- Voicemail server provides the management of voice messages. It manipulates the tasks of handling the database as well as performing encryption of voice messages with AES.

All servers are communicated via the HTTPS to provide voicemail services.

4.3 System Architecture

The architecture of the proposed system as shown in Figure 4.2 consists of three main components. First, user groups are the user interface that accesses the system via a telephone, a soft phone, and a computer as Internet users. The telephone users and soft phone users access to the Asterisk server for call and voicemail services

on SIP channel. The proper configuration is prepared for authorized users. The Internet users access the system via a web browser for administration. In addition, voice messages can be viewed and managed via the web.

The proposed work has two databases: configuration database and voicemail database. The configuration database keeps all the configuration values of Asterisk server and user information. The voicemail database keeps all voice messages. In addition, TLS is used as the main security protocol to protect all messages communicating in the proposed system.

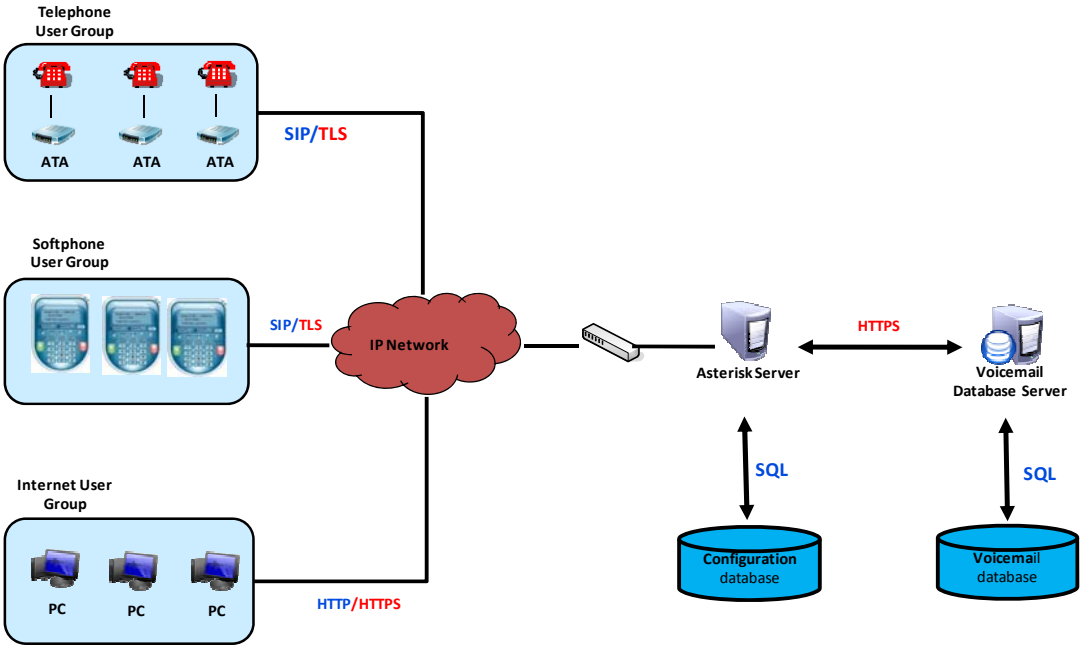


Figure 4.2: System Architecture

4.4 System Model

The proposed model consists of two interfaces, telephone and web interfaces, connecting to the voicemail manager as shown in Figure 4.3. Each component is described in turn as follows.

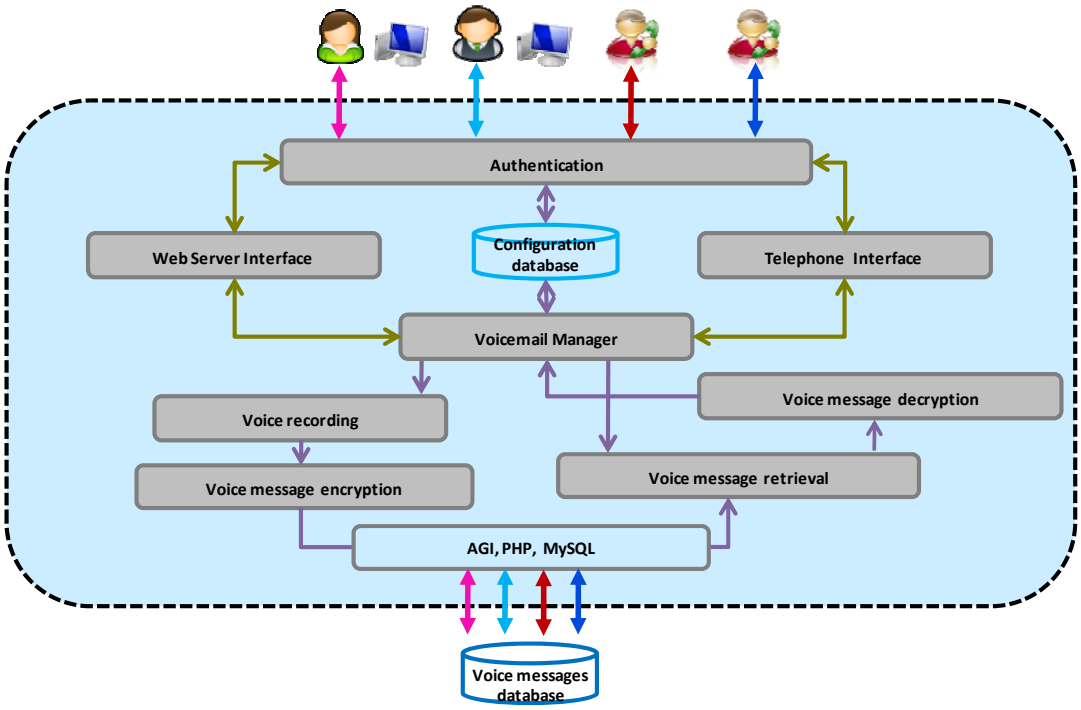


Figure 4.3: System Model for Secure Voicemail Services

At first, users must be authenticated before connecting to the system. Each user can access via telephone or web-based services since the voicemail services are transparent to users. The voicemail manager plays the major role in serving voicemail services since it provides recording and retrieving of the voice messages as well as their encryption and decryption. Asterisk Gateway Interface (AGI) scripts are used to manipulate voicemail database along with PHP scripts and MySQL functions.

4.4.1 Authentication Process

The authentication process has two steps: user registration that includes the certificate request for authorized access to the services, and user authorization via a password as shown in Figure 4.4.

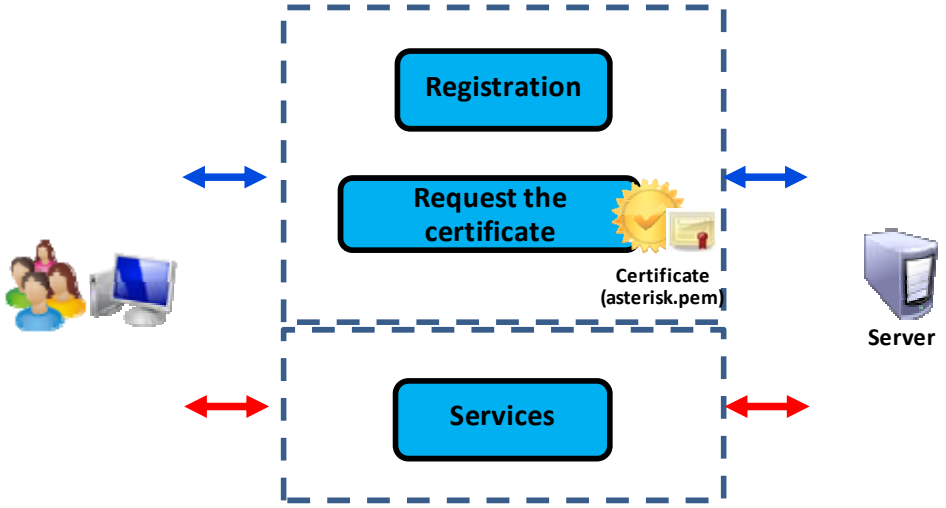


Figure 4.4: Authentication Process

At the registration step, users are requested to fill in the registration form. The system sends an e-mail to the requested user for identification and authorization. When the registration is confirmed, the system activates the user and sends the e-mail with an attached certificate to that user as shown in Figure 4.5. The user can immediately join the system for voicemail services.

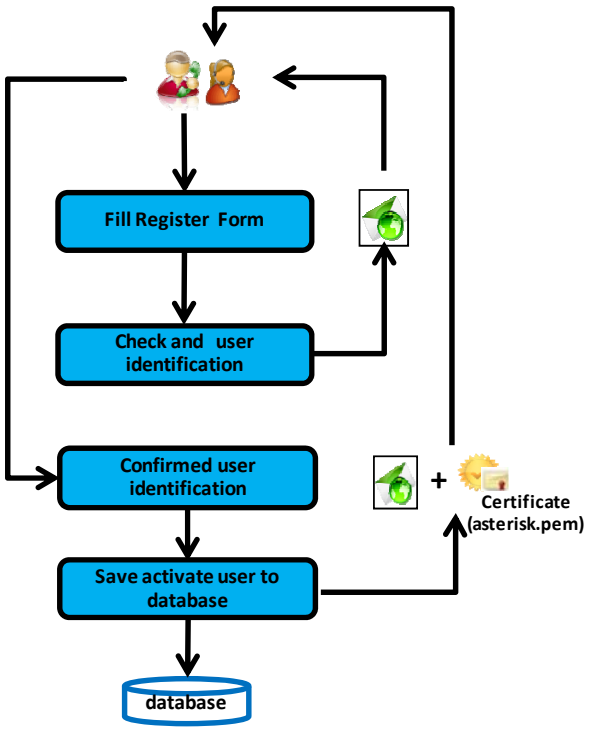


Figure 4.5: Registration Step

4.4.2 Voice Message Encryption and Decryption Process

Figure 4.6 shows the process of voice message encryption and decryption. When a user makes a call and accesses to the voicemail service, the leaving message is processed and sent to the voicemail database server using the AGI script. Then, the voice message is encrypted using the AES-128 algorithm before being saved to the database. In addition, the system calls the PHP script to generate a key for each voice message.

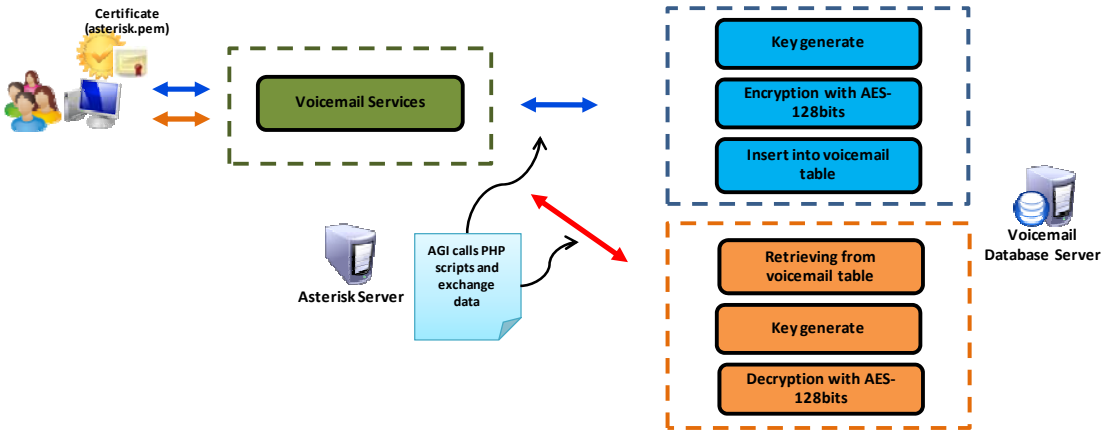


Figure 4.6: Voice Message Encryption and Decryption Process

The voice message decryption process decrypts the retrieving message when the user requests it. The voicemail manager will challenge for a valid password for three times. For the authorized user, the voice message is retrieved from the database and decrypted using the key generation script so that the sound of the decrypted message is returned to the user.

4.4.3 Key Generation

The key generation is provided via a PHP script. It concatenates the data and encrypted using the MD5 function to generate a 128-bit AES key. The key includes the caller id and the voicemail recording date and time as shown in Figure 4.7. This simple idea is defined to protect the voice message in the database.

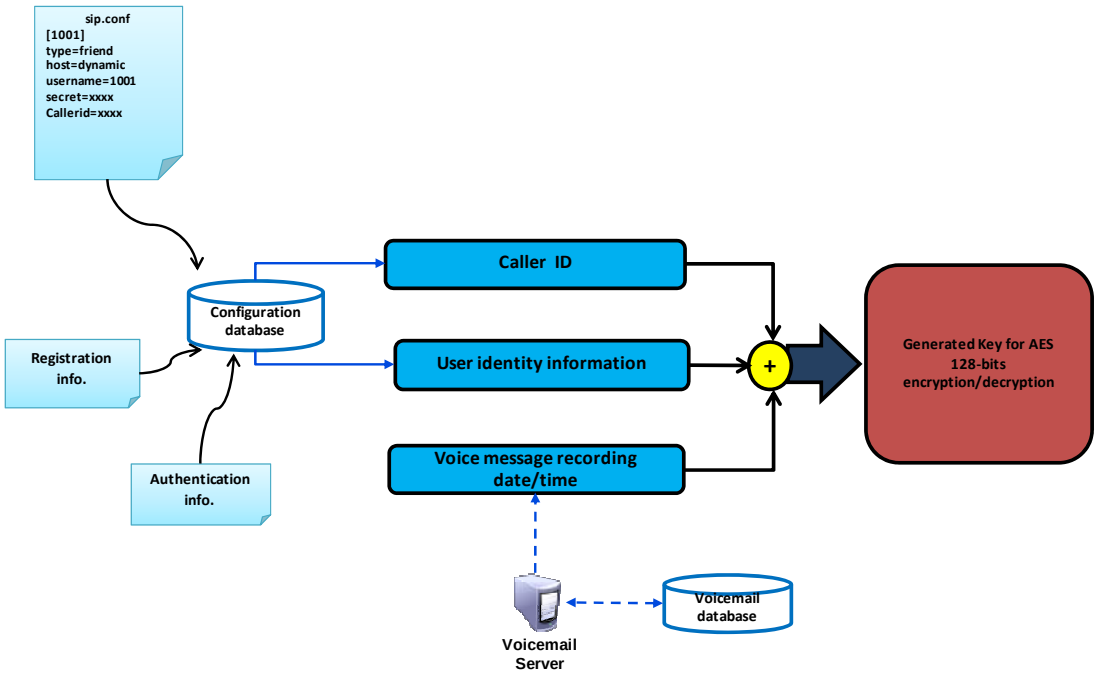


Figure 4.7: Key Generation

CHAPTER V
EXPERIMENTS AND RESULTS

In this chapter, we describe our experiments and the results. We describe the hardware and software configuration, the experiment setup, the performance evaluation, the experimental results and their analysis.

5.1 Hardware and Software Configuration

In this work, our proposed system consists of two components: client nodes and the server node. One client node has a telephone connected via the analogue terminal adapter (ATA) and an ATA Linksys PAP2T to make ordinary calls while another client node is a PC used for a soft phone call. The other PC is used for testing the performance. The server node consists of the Asterisk server and the voicemail database server. The physical overview of the proposed system is shown in Figure 5.1.

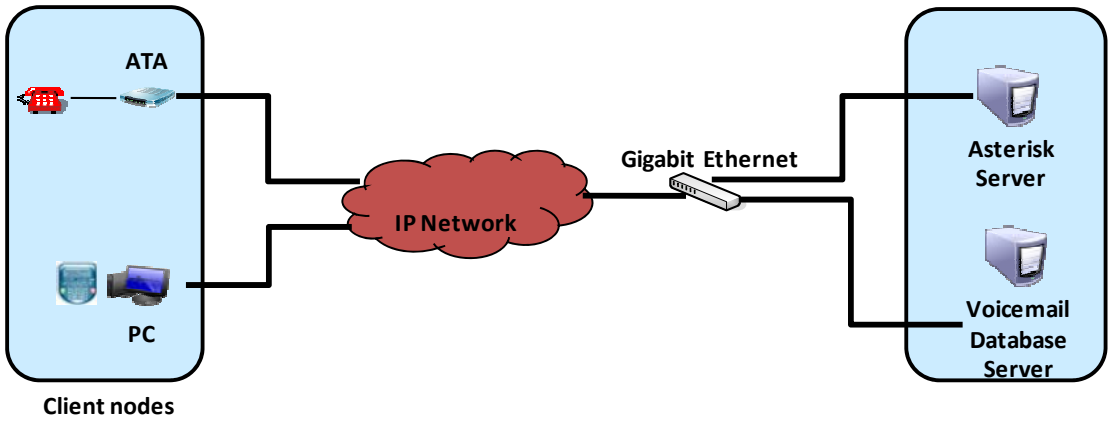


Figure 5.1: Physical Overview of the Proposed System

The proposed system is developed on a FreeBSD operating system having two server nodes having the hardware and software configurations shown in Table 5.1

and 5.2, respectively. The client nodes use the software configuration and the testing tool shown in Table 5.3.

Table 5.1 Hardware Configuration for Server nodes

CPU	Intel(R) Core™2 Quad CPU Q8400 2.66 GHz
Memory	4 GB RAM
Hard disk	320 GB
Network	Gigabit Ethernet

Table 5.2 Software Configuration for Server nodes

OS	FreeBSD 8.0
PBX	Asterisk 1.6.0.15, Asterisk add-ons 1.6.0.3
DBMS	Apache 2.2.13
Web Server	MySQL5.0.86
Test tools	SIPp 3.1, OpenSSL 0.9.8k_5
Programming Language	PHP 5.2.11

Table 5.3 Software Configuration for a PC Client node

OS	Windows XP SP3
Test Tool	SIPp 3.1
Web Browser	IE 6 or later

5.2 Performance Evaluation

In the experiments, we generate VoIP calls and voicemail services with the emphasis on the SIP protocol. The performance evaluation of the system under test can be measured using the standard benchmark known as SPECSIP [http://www.spec.org/specsip/] which consists of the traffic generator and tools to evaluate the system using SIP.

5.2.1 Performance Evaluation Model

The Standard Performance Evaluation Corporation (SPEC) is a non-profit organization formed to establish, maintain and endorse a standardized set of relevant benchmarks applied to evaluate the performance of computers. SPEC develops benchmark suites as well as reviews and publishes submitted results from member organizations and other benchmark licensees. In 2008, the SPEC SIP committee has developed a new industry standard benchmark for evaluating servers using the SIP protocol, and it is called SPECSIP2008 which is used to evaluate the SIP server supporting SIP applications. It also gives guidelines such as use case scenarios tests. Figure 5.2 below shows the call flow of the completed call scenario.

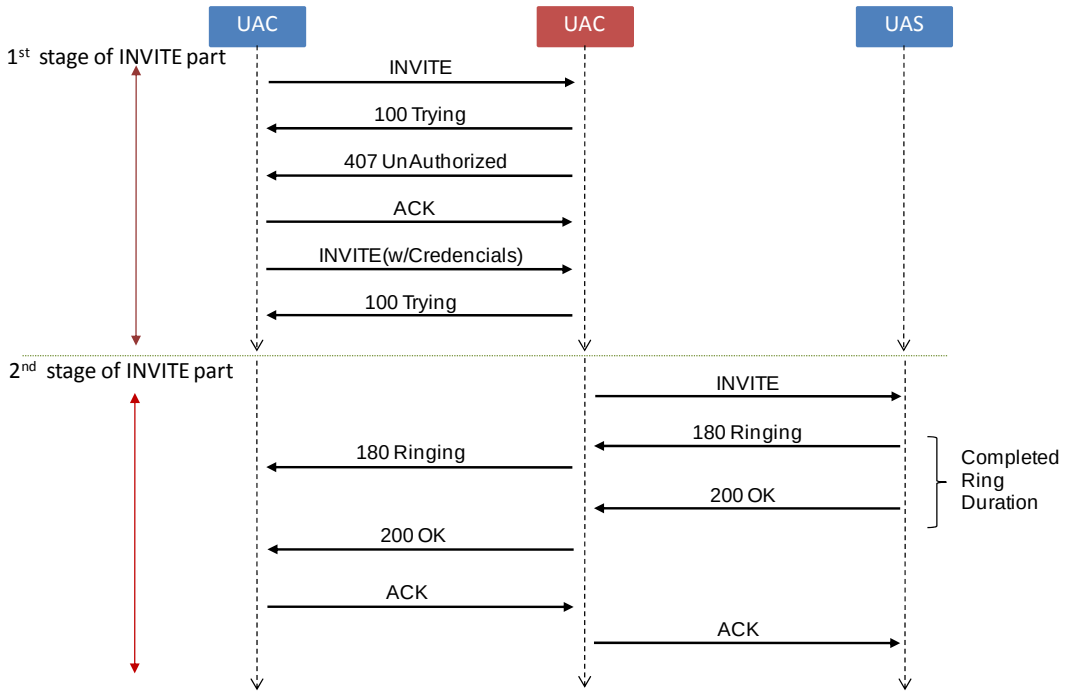


Figure 5.2: A Call Flow of Completed Call Scenario

Figure 5.2 shows that there are two call flows of INVITE parts. The first stage of the INVITE part has the message flows in sequence as follows:

- The UAC sends an initial INVITE request to the SUT.
- The SUT sends a 100 (Trying) response to the UAC.
- The SUT generates a challenge and sends a 407 (Unauthorized) response back to the UAC with the challenge.

- The UAC sends an ACK request for the 407 for completing the first INVITE transaction.
- The UAC generates a credential based on an MD5 hash of the challenge and the user password.
- The UAC then re-transmits the INVITE request to the SUT with an authorization header that includes the credential, creating a new transaction.
- The SUT extracts the public identity, examines the user profile, and verifies the credential.
- The SUT sends a 100 (Trying) response to the UAC.

The call flow of the second stage of the INVITE part proceeds as follows:

- The SUT forwards the authenticated INVITE request to the UAS.
- The UAS sends a 180 (Ringing) response which the SUT forwards back to the UAC.
- The UAS sends a 200 (OK) response which the SUT forwards back to the UAC.
- The UAC sends an ACK request which the SUT forwards to the UAS.

5.2.2 Performance Metrics

In our work, the performance metrics considered are the speed on a VoIP call, the availability of the VoIP system and the response time on a voicemail call. The speed of the system is measured as the time spent to connect, and it is defined as the call setup delay. The availability is the number of calls that can be served by the system, and it is measured by analyzing the failed call rates. For the voicemail call, the response time spent for listening voice message on a retrieving voice message process is measured.

- **Call Setup Delay**

The call setup delay is defined as the interval between entering the last dialed digit and receiving a ring. In this work, it is measured in milliseconds from

sending the first SIP message of a call until receiving a response of the 200 OK message.

- **Failed Call Rates**

The statistics of failed call rates when there are many concurrent calls can be regarded as the effective throughput of the system. A call is defined as a failed call when the following event has occurred.

- FailedCannotSendMessage: the number of failed calls because SIP cannot send the message due to transport issues.
- FailedMaxUDPRetrans: the number of failed calls because the maximum number of UDP retransmission attempts has been reached.
- FailedUnexpectedMessage: the number of failed calls because the SIP message received is not expected in the scenario.
- FailedCallRejected: the number of failed calls because of SIP internal errors. For example, a scenario sync command is not recognized or a scenario action failed or a scenario variable assignment failed.
- FailedCmdNotSent: the number of failed calls because of inter-SIP communication errors such as a scenario sync command failed to be sent.
- FailedTimeoutOnRecv: the number of failed calls because of a receiving timeout statement.
- FailedTimeoutOnSend: the number of failed calls because of a sending timeout statement.

- **Voicemail Call**

The voicemail call is defined to measure the response time at the period of the voice message retrieving process. It starts when a caller sending an order to listen to the voice message until receiving this voice message as shown in Figure 5.3. The response time is calculated by including the process time beginning when the Asterisk server sending a request to the voicemail database server (X), the process time to retrieve the request data from the database (Y), and the time used to return the request data to the Asterisk server (Z).

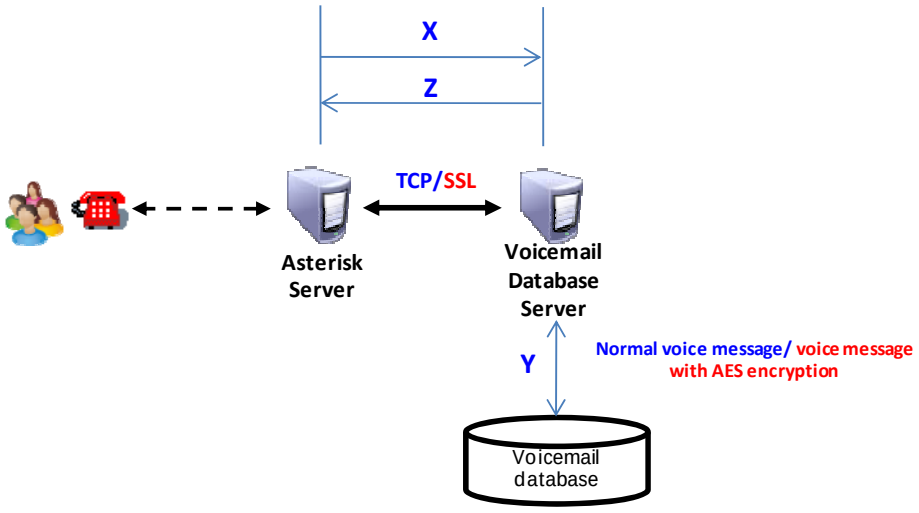


Figure 5.3: Response Time of Voicemail Retrieving Process

5.3 Experimental Setup

Two main systems are setup for this experiment to compare the performance. They are the baseline system and the proposed system. The baseline system is configured as the traditional Asterisk PBX. The proposed system is configured to improve the security mechanisms as previously mentioned.

5.3.1 Test Bed Systems

Figures 5.4 and 5.5 show the test bed system of the baseline system and the test bed system of the proposed system, respectively.

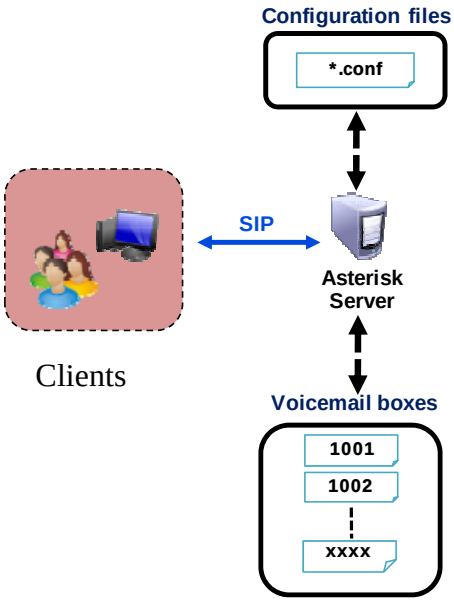


Figure 5.4: Baseline Test Bed System

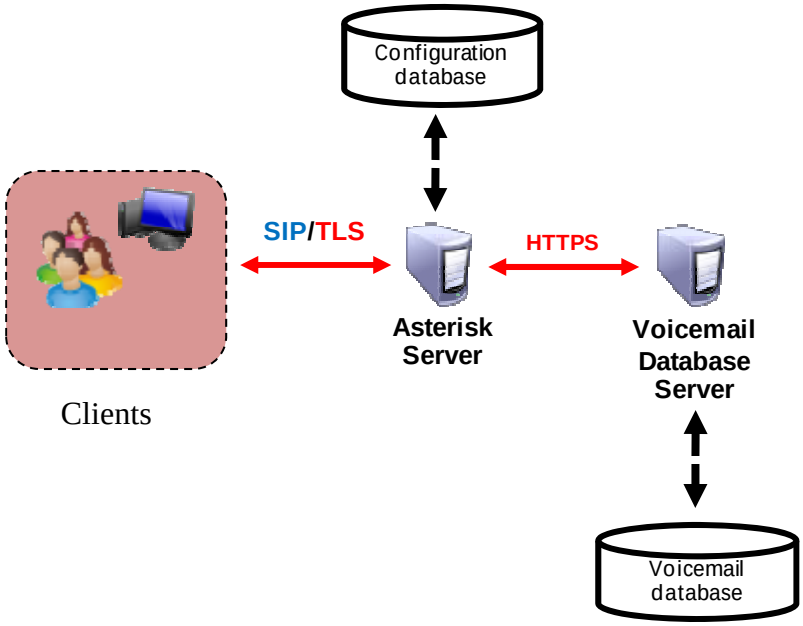


Figure 5.5: Proposed Test Bed System

5.3.2 Test Bed Configuration

The configuration of both test bed systems is assigned as shown in Table 5.4.

Table 5.4: Test Bed Configuration

Asterisk configuration	*.conf files	Database
Voicemail box	Directory	Database
Voice message	No encrypted	AES encrypted
Signaling/Transport protocol	SIP/UDP	SIP/TLS

The Asterisk configuration setup of call setup delay and availability are given in detail below.

- **sip.conf**

  [sipp]
  type=friend
  host=dynamic
  username=sipp
  context=sip

- **extensions.conf**

  [sipp]
  exten => 2005,1,Answer
  exten => 2005,n,Dial(SIP/2005,40,r)
  exten => 2005,n,Voicemail(2005@default,b)
  exten => 2005,n,SetMusicOnHold(default)
  exten => 2005,n,WaitMusicOnHold
  exten => 2005,n,Hangup
  [default]
  exten => 8888,1,Answer
  exten => 8888,n,VoiceMailMain()
  exten => 8888,n,Hangup

- **voicemail.conf**

  [default]
  2005 => 2005,2005,asterisk@asterisk.mahidol.ac.th

The additonal Asterisk configuration setup of voicemail call is given in detail below.

- **extensions.conf**

```
[default]
exten  =>  9000,1,Answer
; playback Commedian Mail. Mailbox: message
exten  => 9000,2,Playback(vm-login)
;Wait for mailbox number
exten  => 9000,3,Wait(3)
; playback password message
exten  => 9000,4,Playback(vm-password)
;Wait for enter password
exten  => 9000,5,Wait(3)
; playback You have message
exten  => 9000,6,Playback(vm-youhave)
; play a number of messages in mailbox
exten  => 9000,7,SayDigits(1)
; playback message word
exten  => 9000,8,Playback(vm-message)
;playback press 1 to listen to message
exten  => 9000,9,Playback(vm-onefor-full)
;retrieving voice message from voicemail database server
exten  => 9000,10,AGI(vmget.php)
; Playback voice message to caller
exten  => 9000,11,Playback(/tmp/vm0000.WAV)
 exten => 9000,n,Hangup
```

5.3.3 Test Bed Scenario

A call scenario such as the completed call scenario is used to integrate many scenarios into invite.xml files. The call flow of the test bed scenario is shown in Figure 5.6.

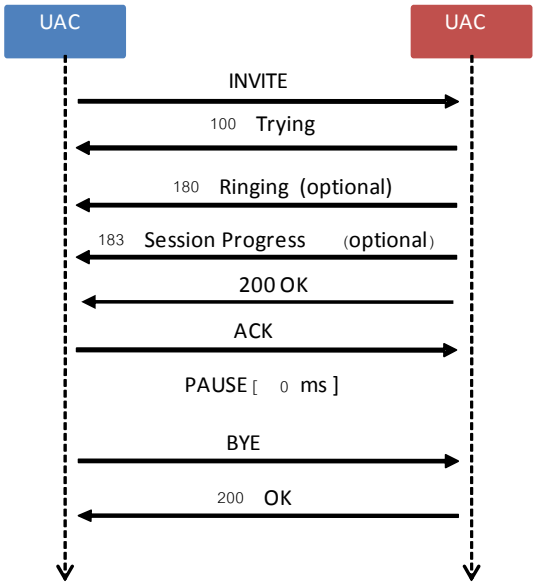


Figure 5.6: Call Flow of Test Bed Scenario

5.3.4 Testing Tools

Two testing tools: SIPp and OpenSSL are used for evaluating the performance and they are described below.

- **SIPp [24]** is a free open source traffic generator for the SIP protocol. It supports various features including a simple flow control, integrated scenarios such as reading custom XML scenario files describing from very simple to complex call flows, multiple transport modes such as UDP, TCP, and TLS over multiple sockets or multiplexed with retransmission management and dynamically adjustable call rates. Its features include the dynamic display of statistics of running tests (call rates, round trip delay, and message statistics), and periodic statistics reports.
- **OpenSSL [26]** is the open source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-fledged cryptography library. It is used to create a self-sign certificate for the TLS support of the proposed work.

5.4 Experiments and Results

We use SIPp to generate calls to test the experiments by reading *invite.xml* scenario as shown in Figure 5.6. The number of concurrent calls ranges from 50, 100, 150 till 500. Each set runs for 120 seconds for three times. The resulting response times are collected as intervals in milliseconds and their ranges of intervals are shown in Table 5.5 where n is the average response time.

Table 5.5 Range of Interval Response Time

0ms<= n < 10ms
10ms<= n < 20ms
20ms<= n < 30ms
30ms<= n < 40ms
…
180ms<= n < 190ms
190ms<= n < 200ms
200ms<= n < 300ms
…
900ms<= n < 1000ms
1000ms<= n < 2000ms
2000ms<= n < 3000ms
…
9000ms<= n < 10000ms
n <= 10000ms

For supporting TLS, Asterisk requires a certificate expressed in the asterisk.pem file which comprises of two parts: a private key and a certificate. In this work, we use the OpenSSL to create both as described in detail below.

1. Generate a private RSA key of 1024 bits in the output file of key.pem via the following command:

   # openssl genrsa –out key.pem 1024

2. Create a certificate in the request.pem file, using the private key already built in the file, key.pem, via the following command:

   # openssl req –new –key key.pem –out request.pem

3. Self-sign a certificate request and keep in the file, certificate.pem, via the following command:

   # openssl x509 –req –days 365 –in request.pem –signkey key.pem –out certificate.pem

4. Create an asterisk.pem that combines both the private key and the certificate into the same file via the following command:

   # cat key.pem > asterisk.pem

   # cat certificate.pem >> asterisk.pem

5.4.1 Call Setup Delay Experiments on Baseline System

This experiment measures the call setup delay and failed call rates on the baseline system. The command format and the assigned parameters are given as follows.

 # sipp -sf <scenario name> **-s** <extensions> **-fd** 1 **-l** <Number of concurrent calls> **-r** <Number of concurrent calls> **-rp** <assigned time in millisecond> **-i** <local ip address:port> <asterisk server ip address:port> **-trace_screen -trace_err -trace_stat -trace_rtt**

The details of each command option of SIPp are given in Table 5.6 below. The test is run and the outputs are collected from the dump files containing the results. The sample screen captures of the experiments are shown in Figures 5.7, 5.8 and 5.9.

Table 5.6: Option Parameters of SIPp

Parameter options	**Description**
-sf	Load an alternate xml scenario file.
-s	Set the username part of the request URI. Default is 'service'.
-fd	Set the statistics dump log report frequency. Default is 60 and default unit is seconds.
-l	Set the maximum number of simultaneous calls. Default: (3 * call_duration (s) * rate).
-r	Set the call rate (# calls per seconds). If the -rp option is used, the call rate is calculated with the period in ms given by the user.
-rp	Specify the rate period for the call rate. Default is 1 second and default unit is milliseconds. This allows you to have n calls every m milliseconds (by using -r n -rp m). Example: -r 7 -rp 2000 ==> 7 calls every 2seconds. -r 10 -rp 5s ==> 10 calls every 5 seconds.
-i	Set the local IP address for 'Contact:','Via:', and 'From:' headers. Default is the primary host IP address.
-trace_screen	Dump statistic screens in the <scenario_name>_<pid>_screen.log file when quitting SIPp.
-trace_err	Trace all unexpected messages in <scenario file name>_<pid>_errors.log.
-trace_stat	Dumps all statistics in <scenario_name>_<pid>.csv file. Use the '-h stat' option for a detailed description of the statistics file content. Allow tracing of all response times in <scenario file name>_<pid>_rtt.csv.
-trace_rtt	Allow tracing of all response times in <scenario file name>_<pid>_rtt.csv.

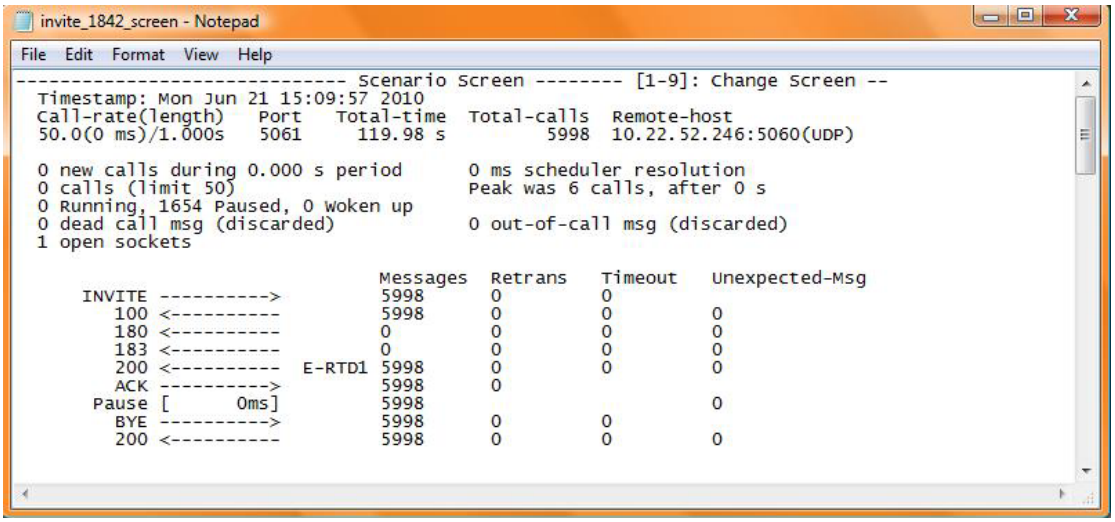


Figure 5.7: Sample Scenario Screen Capture

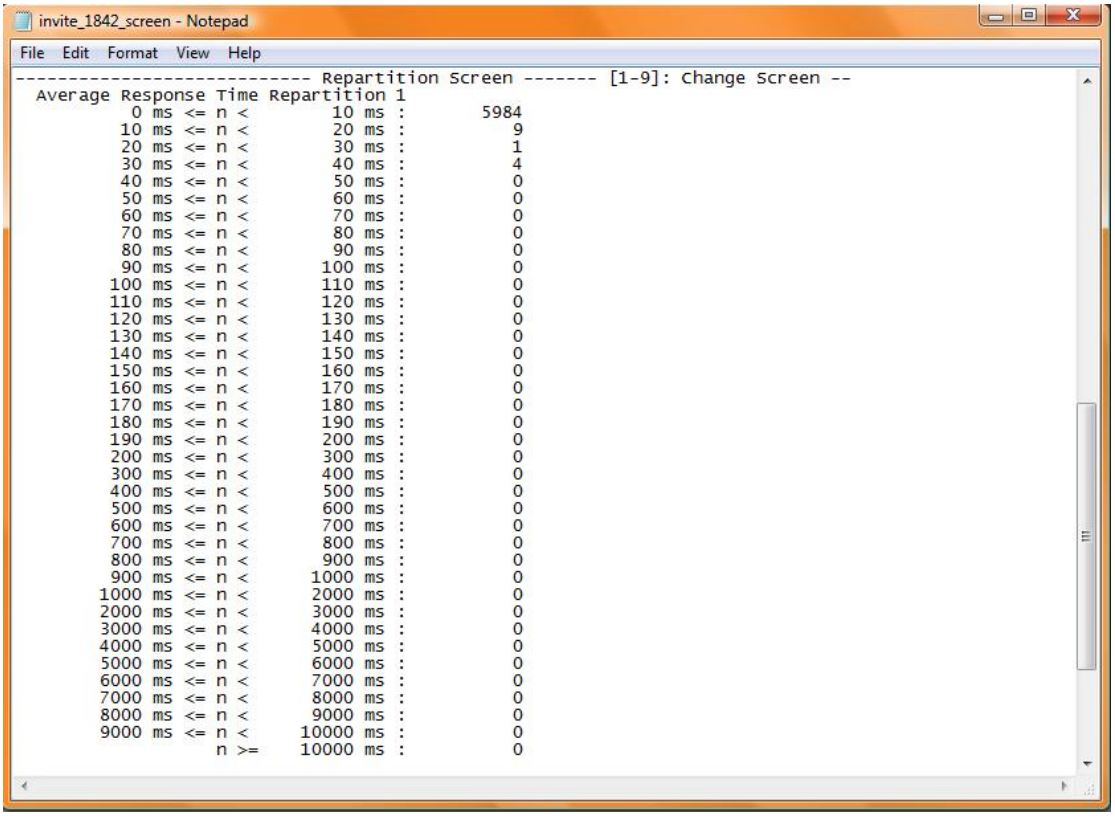


Figure 5.8: Sample Repartition Screen Capture

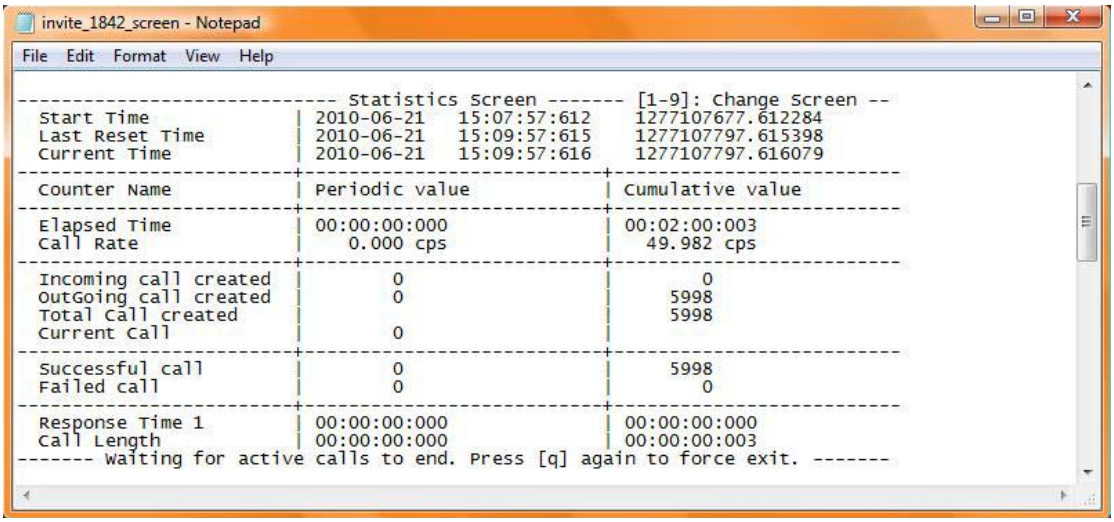


Figure 5.9: Sample Statistics Screen Capture

Table 5.7 shows the number of concurrent calls in percentage on each response time range.

Table 5.7: Percentage of Concurrent Calls in Interval Response Time Ranges

Range of Interval Response Time	Concurrent calls		Range of Interval Response Time	Concurrent calls	
	50	500		50	500
0ms<= n < 10ms	99.17	0.00	110ms<= n < 120ms	0.00	1.46
10ms<= n < 20ms	0.83	0.21	120ms<= n < 130ms	0.00	6.17
20ms<= n < 30ms	0.00	0.00	130ms<= n < 140ms	0.00	0.00
30ms<= n < 40ms	0.00	0.26	140ms<= n < 150ms	0.00	7.09
40ms<= n < 50ms	0.00	0.53	150ms<= n < 160ms	0.00	7.70
50ms<= n < 60ms	0.00	0.00	160ms<= n < 170ms	0.00	0.00
70ms<= n < 80ms	0.00	1.74	170ms<= n < 180ms	0.00	6.75
80ms<= n < 90ms	0.00	2.92	180ms<= n < 190ms	0.00	6.41
90ms<= n < 100ms	0.00	0.00	190ms<= n < 200ms	0.00	0.00
100ms<= n < 110ms	0.00	4.48	200ms<= n < 300ms	0.00	17.24

Range of Interval Response Time	Concurrent calls		Range of Interval Response Time	Concurrent calls	
	50	500		50	500
300ms<= n < 400ms	0.00	1.74	3000ms<= n < 4000ms	0.00	5.73
400ms<= n < 500ms	0.00	0.67	4000ms<= n < 5000ms	0.00	0.30
500ms<= n < 600ms	0.00	0.26	5000ms<= n < 6000ms	0.00	0.03
600ms<= n < 700ms	0.00	5.01	6000ms<= n < 7000ms	0.00	0.10
700ms<= n < 800ms	0.00	3.85	7000ms<= n < 8000ms	0.00	4.28
800ms<= n < 900ms	0.00	0.39	8000ms<= n < 9000ms	0.00	0.05
900ms<= n < 1000ms	0.00	0.16	9000ms<= n < 10000ms	0.00	0.07
1000ms<= n < 2000ms	0.00	8.01	n >= 10000 ms	0.00	2.43
2000ms<= n < 3000ms	0.00	0.22			

Table 5.8 shows the percentage of successful calls and the failed call rates for each concurrent call's range.

Table 5.8: Successful Call and Failed Call Rates for Each Concurrent Call Range

5.4.2 Call Setup Delay Experiments on the Proposed System

This experiment measures the call setup delay and failed call rates on the proposed system. The command format and the assigned parameters are given as follows.

 # sipp -sf <scenario name> **-s** <extensions> **-fd** 1 **-l** <Number of concurrent calls> **-r** <Number of concurrent calls> **-rp** <assigned time in millisecond> **-i** <local ip address:port> <asterisk server ip address:port> **-trace_screen -trace_err -trace_stat -trace_rtt -tls_cert** asterisk.pem **-tls_key** asterisk.pem **-max_reconnect** <int value> **-reconnect_close** true **-reconnect_sleep** <int value> **-t** ln

The details of additional command options of SIPp when TLS is used are given in Table 5.9 below. The test is run and the outputs are collected from the dump files containing the results. The sample screen captures of the experiments are shown in Figures 5.10, 5.11 and 5.12.

Table 5.9 Additional Options Parameter of SIPp on TLS

Parameter options	Description
-tls_cert	Set the name for TLS Certificate file. Default is 'cacert.pem
-tls_key	Set the name for TLS Private Key file. Default is 'cakey.pem'
-t	Set the transport mode: - u1: UDP with one socket (default), - un: UDP with one socket per call, - ui: UDP with one socket per IP address The IP address must be defined in the injection file. - t1: TCP with one socket, - tn: TCP with one socket per call, - l1: TLS with one socket, - ln: TLS with one socket per call, - c1: u1 + compression (only if compression plug in loaded), - cn: un + compression (only if compression plug in loaded). This plug in is not provided with SIPp.
-max_reconnect	Set the maximum number of reconnection attempts.
-reconnect_close	Should calls be closed on reconnect?
-reconnect_sleep	int: How long to sleep (in milliseconds) between the close and reconnect?

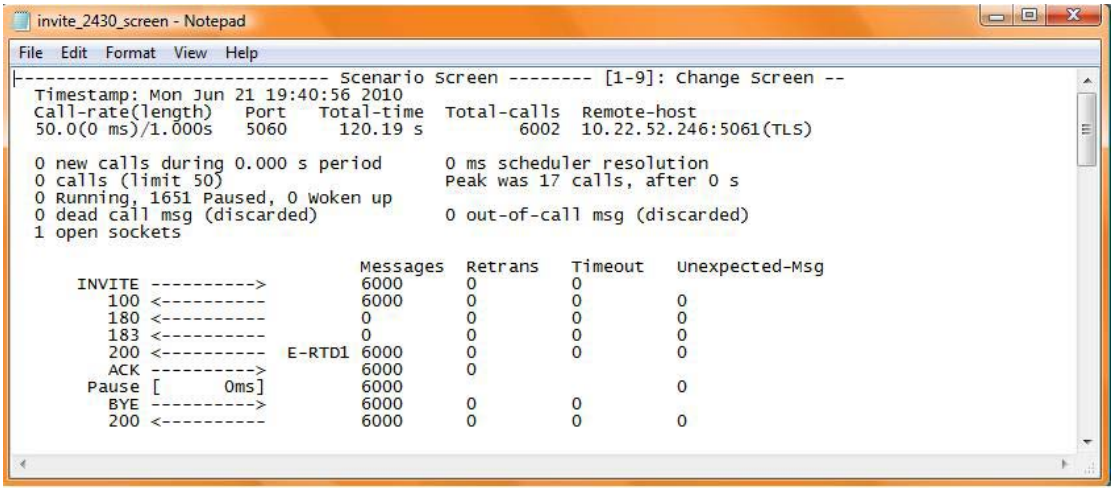
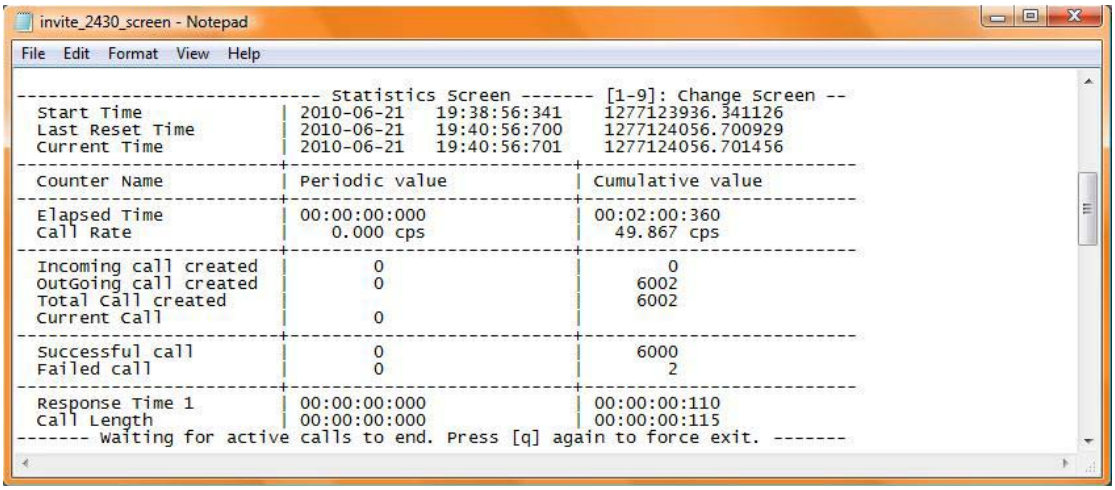


Figure 5.10: Sample Scenario Screen Capture

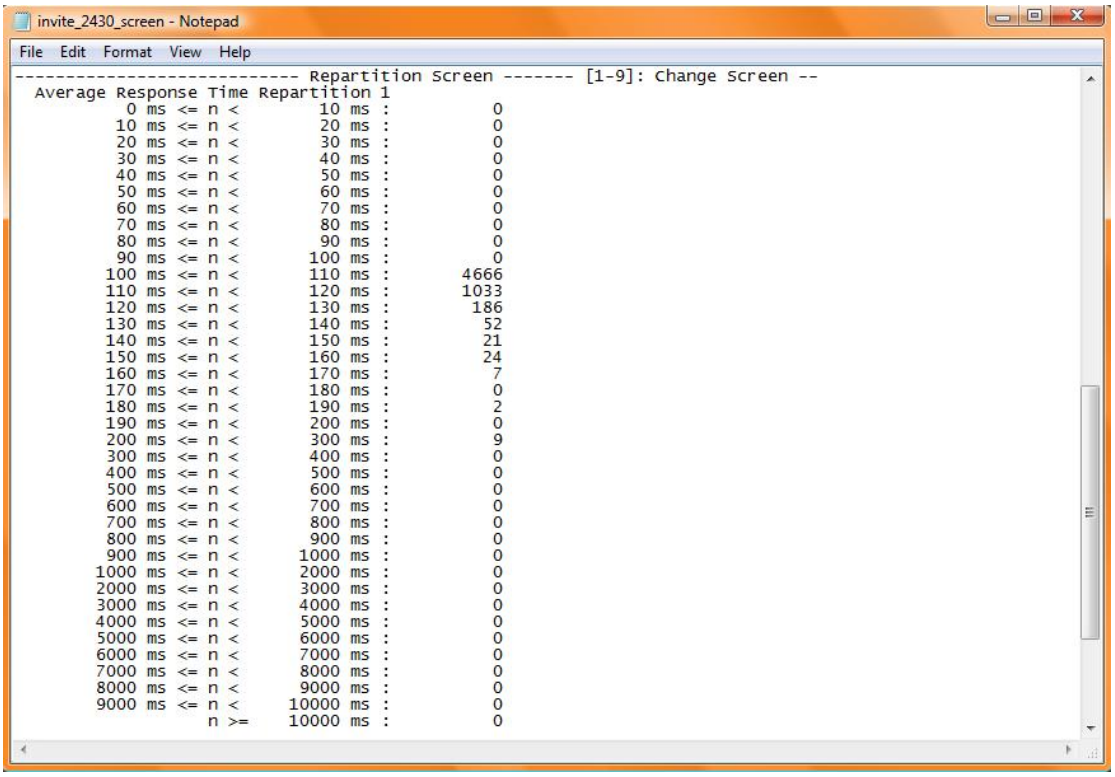


invite_2430_screen - Notepad

File Edit Format View Help

```
------------------------------ Statistics Screen ------- [1-9]: Change Screen --
  Start Time                | 2010-06-21   19:38:56:341    1277123936.341126
  Last Reset Time           | 2010-06-21   19:40:56:700    1277124056.700929
  Current Time              | 2010-06-21   19:40:56:701    1277124056.701456
----------------------------+-------------------------+-------------------------
  Counter Name              | Periodic value          | Cumulative value
----------------------------+-------------------------+-------------------------
  Elapsed Time              | 00:00:00:000            | 00:02:00:360
  Call Rate                 |    0.000 cps            |   49.867 cps
----------------------------+-------------------------+-------------------------
  Incoming call created     |        0                |        0
  OutGoing call created     |        0                |     6002
  Total Call created        |                         |     6002
  Current Call              |        0                |
----------------------------+-------------------------+-------------------------
  Successful call           |        0                |     6000
  Failed call               |        0                |        2
----------------------------+-------------------------+-------------------------
  Response Time 1           | 00:00:00:000            | 00:00:00:110
  Call Length               | 00:00:00:000            | 00:00:00:115
------- Waiting for active calls to end. Press [q] again to force exit. -------
```

Figure 5.11 Sample Statistics Screen Capture



invite_2430_screen - Notepad

File Edit Format View Help

```
---------------------------- Repartition Screen ------- [1-9]: Change Screen --
  Average Response Time Repartition 1
          0 ms <= n <        10 ms :          0
         10 ms <= n <        20 ms :          0
         20 ms <= n <        30 ms :          0
         30 ms <= n <        40 ms :          0
         40 ms <= n <        50 ms :          0
         50 ms <= n <        60 ms :          0
         60 ms <= n <        70 ms :          0
         70 ms <= n <        80 ms :          0
         80 ms <= n <        90 ms :          0
         90 ms <= n <       100 ms :          0
        100 ms <= n <       110 ms :       4666
        110 ms <= n <       120 ms :       1033
        120 ms <= n <       130 ms :        186
        130 ms <= n <       140 ms :         52
        140 ms <= n <       150 ms :         21
        150 ms <= n <       160 ms :         24
        160 ms <= n <       170 ms :          7
        170 ms <= n <       180 ms :          0
        180 ms <= n <       190 ms :          2
        190 ms <= n <       200 ms :          0
        200 ms <= n <       300 ms :          9
        300 ms <= n <       400 ms :          0
        400 ms <= n <       500 ms :          0
        500 ms <= n <       600 ms :          0
        600 ms <= n <       700 ms :          0
        700 ms <= n <       800 ms :          0
        800 ms <= n <       900 ms :          0
        900 ms <= n <      1000 ms :          0
       1000 ms <= n <      2000 ms :          0
       2000 ms <= n <      3000 ms :          0
       3000 ms <= n <      4000 ms :          0
       4000 ms <= n <      5000 ms :          0
       5000 ms <= n <      6000 ms :          0
       6000 ms <= n <      7000 ms :          0
       7000 ms <= n <      8000 ms :          0
       8000 ms <= n <      9000 ms :          0
       9000 ms <= n <     10000 ms :          0
                n >=      10000 ms :          0
```

Figure 5.12: Sample Repartition Screen Capture

Table 5.10 shows the number of concurrent calls in percentage on each response time range.

Table 5.10: Percentage of Concurrent Calls in Interval Response Time Ranges

Range of Interval Response Time	Concurrent calls		Range of Interval Response Time	Concurrent calls	
	50	500		50	500
0ms<= n < 10ms	0.00	0.00	110ms<= n < 120ms	99.19	0.00
10ms<= n < 20ms	0.00	0.00	120ms<= n < 130ms	0.81	0.00
20ms<= n < 30ms	0.00	0.00	130ms<= n < 140ms	0.00	0.00
30ms<= n < 40ms	0.00	0.00	140ms<= n < 150ms	0.00	0.00
40ms<= n < 50ms	0.00	0.00	150ms<= n < 160ms	0.00	0.00
50ms<= n < 60ms	0.00	0.00	160ms<= n < 170ms	0.00	0.00
70ms<= n < 80ms	0.00	0.00	170ms<= n < 180ms	0.00	0.00
80ms<= n < 90ms	0.00	0.00	180ms<= n < 190ms	0.00	0.04
90ms<= n < 100ms	0.00	0.00	190ms<= n < 200ms	0.00	0.01
100ms<= n < 110ms	0.00	0.00	200ms<= n < 300ms	0.00	0.00

Range of Interval Response Time	Concurrent calls		Range of Interval Response Time	Concurrent calls	
	50	500		50	500
300ms<= n < 400ms	0.00	0.27	3000ms<= n < 4000ms	0.00	16.97
400ms<= n < 500ms	0.00	0.40	4000ms<= n < 5000ms	0.00	0.00
500ms<= n < 600ms	0.00	0.96	5000ms<= n < 6000ms	0.00	0.00
600ms<= n < 700ms	0.00	0.99	6000ms<= n < 7000ms	0.00	0.00
700ms<= n < 800ms	0.00	2.54	7000ms<= n < 8000ms	0.00	0.00
800ms<= n < 900ms	0.00	7.21	8000ms<= n < 9000ms	0.00	0.00
900ms<= n < 1000ms	0.00	7.48	9000ms<= n < 10000ms	0.00	0.00
1000ms<= n < 2000ms	0.00	6.86	n >= 10000 ms	0.00	0.00
2000ms<= n < 3000ms	0.00	56.26			

Table 5.11 shows the percentage of successful calls and the failed call rates for each concurrent call's range.

Table 5.11: Successful Call and Failed Call Rates for Each Concurrent Call Range

Concurrent calls	**Successful Call**	**Failed Call**
50	100.00	0.00
60	100.00	0.00
70	99.83	0.07
80	99.79	0.10
90	99.87	0.03
100	50.06	48.80
110	33.54	65.64
120	30.19	68.91
130	25.54	73.15
140	23.98	74.61
150	18.54	80.18

5.4.3 Voicemail Call Experiments

In this experiment, there are four conditions to test. The objective is to compare the process time of retrieving normal and unencrypted voice messages with that of retrieving AES encrypted voice messages based on the general protocol, TCP, and the security protocol, SSL. Each condition was run for five times and the results are the average response time. There are three types of voice message test cases according to the duration of each message which has the length of 10 seconds, 30 seconds and 60 seconds. All voice messages are encoded in gsm format file type which is the standard used on Asterisk.

All voice messages are stored in the MySQL database by reading the entire file into a string before inserting it into the database table. The AES key is generated as the output of the md5 function when concatenating the assigned constant variable, such as the Asterisk server name, and the voicemail properties stored in the database at the same record such as callerid (Caller ID) and origtime (the time of leaving a voice message). This generated key is used as the parameter of the AES_ENCRYPT function for encrypting leaved voice message and AES_DECRYPT function for decrypting retrieved voice message using the SQL command in the PHP script. During the retrieving process, the returned data must write a string to the file before playing this file to the caller. If the generated AES key is wrong, the voice message file cannot be played, and no voice is generated to the caller.

The SSL connection of the Apache web server between Asterisk server and Voicemail database server requires an SSL certificate as the server.crt file and SSL certificate key as the server.key file. In this work, we use the OpenSSL to create them and make the configuration on the Apache web server to activate the SSL connection as described in detail below.

1. Generate a private RSA key of 1024 bits in the output file of server.key via the command:

   ```
   # openssl genrsa –out server.key 1024
   ```

2. Create a certificate in the server.csr file using the private key, server.key, via the command:

   ```
   # openssl req –new –key server.key –out server.csr
   ```

3. Self-sign a certificate request and keep in the file, server.crt, via the command:

   # openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt

4. Move both server.crt and server.key file to the configured location, via the command:

   # cp server.crt /usr/local/etc/apache22/

   # cp server.key /usr/local/etc/apache22/

5. Edit httpd.conf to setup SSL connection by remove # before the line as “include etc/apache22/extra/httpd-ssl.conf”.
6. Recheck the specify location of the server.crt and server.key is right on the httpd-ssl.conf file at line as

   SSLCertificateFile /usr/local/etc/apache22/server.crt

   SSLCertificateKeyFile /usr/local/etc/apache22/server.key

7. Insert “ apache22_flags="-DSSL" ” to the rc.conf file
8. Restart httpd to active SSL service, using command

   # /usr/local/etc/rc.d/apache22 restart

The results of this experiment for each assigned condition are given in Table 5.12.

Table 5.12: Average Response Time of Retrieving Voice Messages

5.5 Analysis of Experimental Results

The results of our experiments on the two systems are explained and discussed as follows.

5.5.1 Call Setup Delay Analysis

Figures 5.13 and 5.14 show and compare the results of the average response time obtained on the baseline system. The response time is measured for 50 and 500 concurrent calls. Obviously, the response times of the two cases are different in a way that most of the response times in 50 concurrent calls are less than 30 ms whereas most of the response times in 500 concurrent calls are almost evenly distributed from 10 to 10000 ms. Figure 5.15 compares the two results on the same scale.

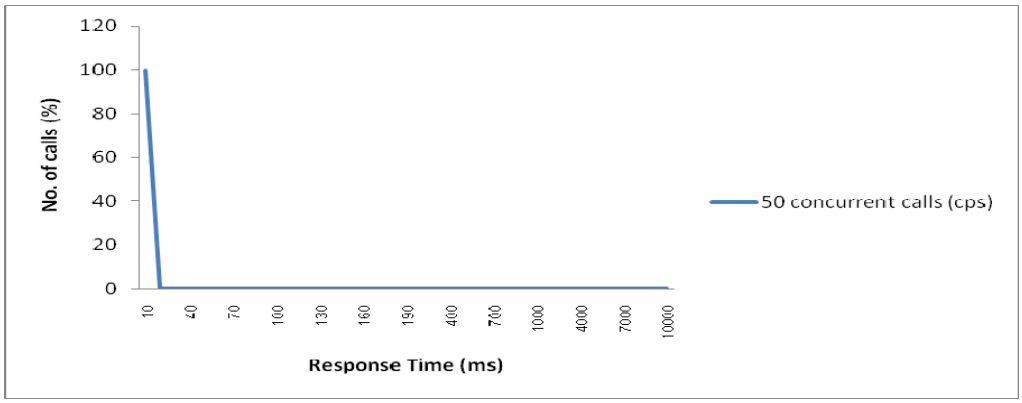


Figure 5.13: Response Time Distribution of 50 Concurrent Calls on Baseline System

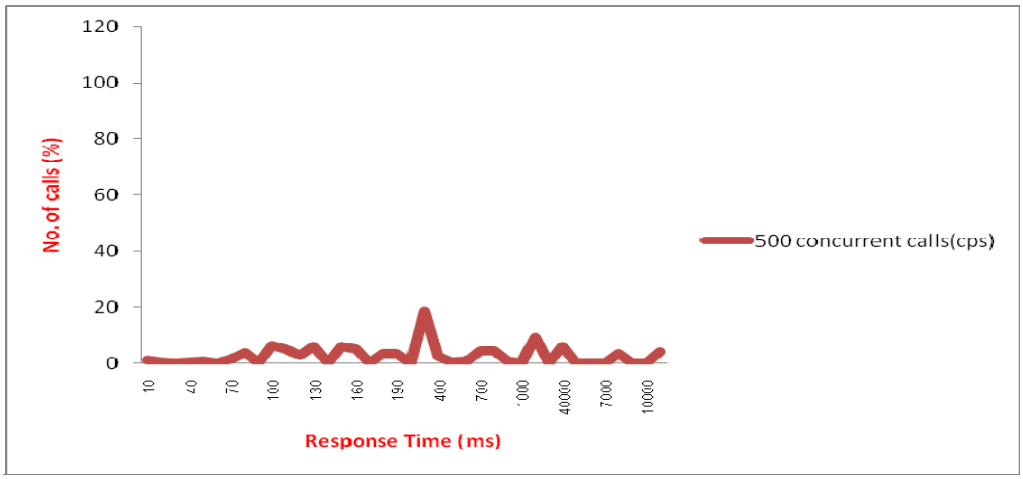


Figure 5.14: Response Time Distribution of 500 Concurrent Calls on Baseline System

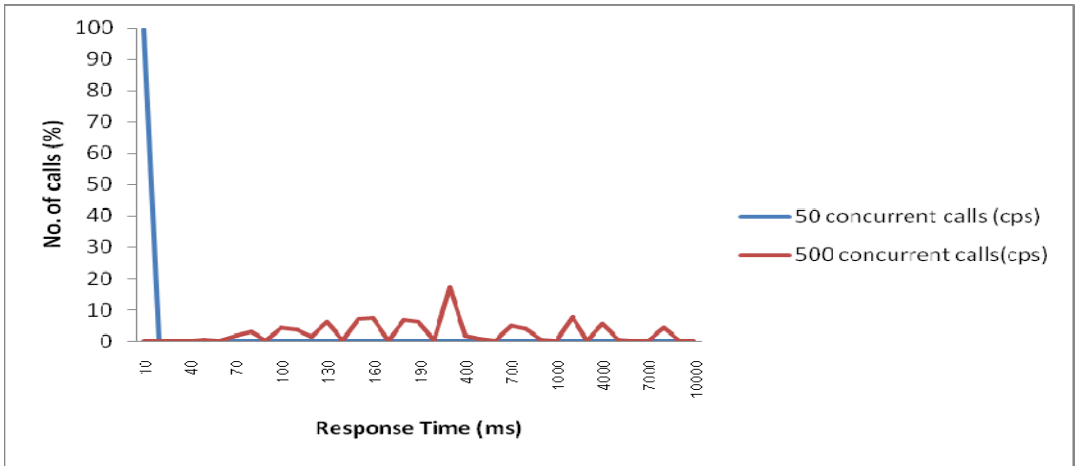


Figure 5.15: Compared Response Times of Baseline System

Figures 5.16 and 5.17 show and compare the results of the response time obtained on the proposed system. The response time is also measured for 50 and 500 concurrent calls. Obviously, the response times of the two cases are different in a way that the response times in 50 concurrent calls has the high peak around 100-110 ms whereas the response times in 500 concurrent calls has the high peak around 1000-4000 ms. Figure 5.18 compares the two results on the same scale.

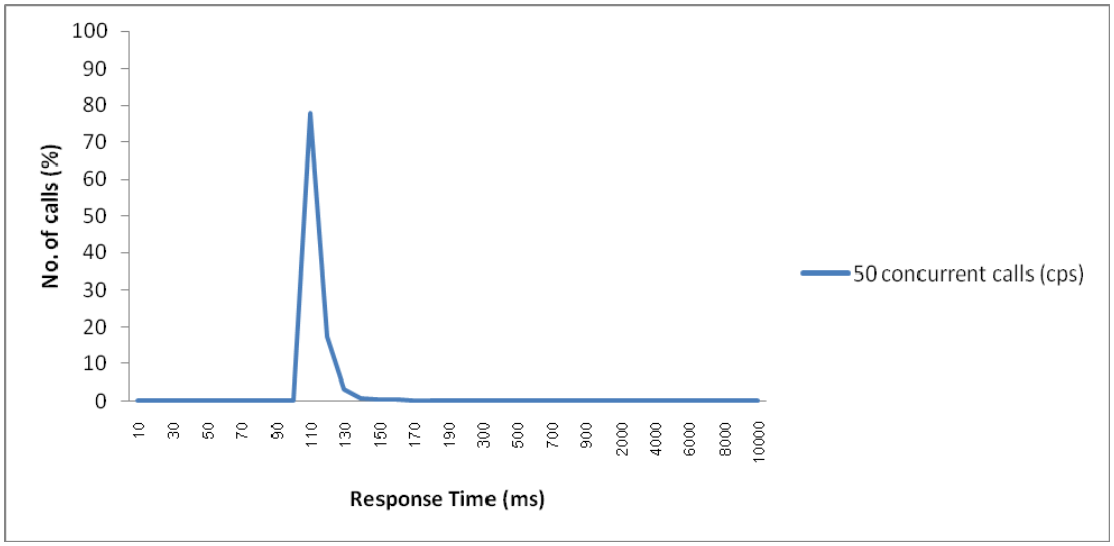


Figure 5.16: Response Time Distribution of 50 Concurrent Calls on Proposed System

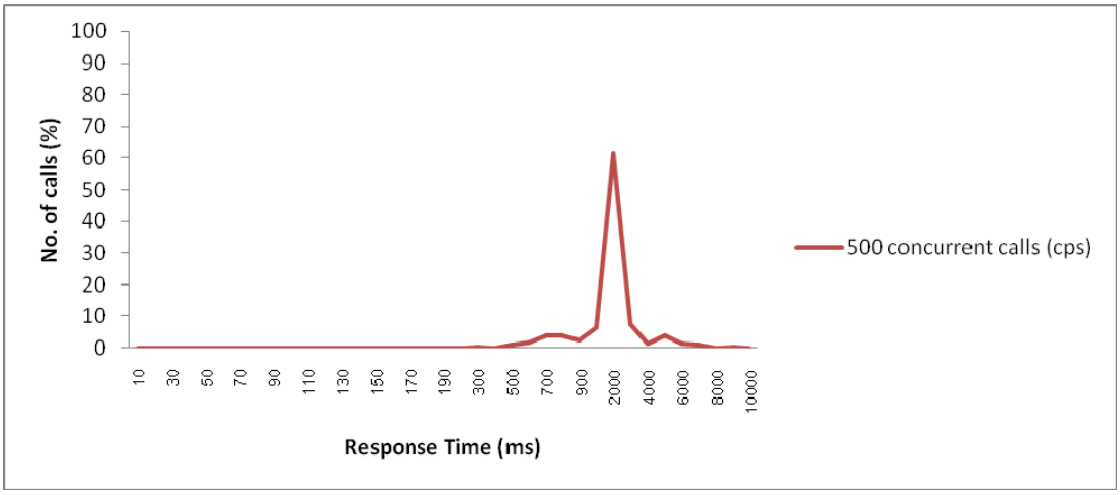


Figure 5.17: Response Time Distribution of 500 Concurrent Calls on Proposed System

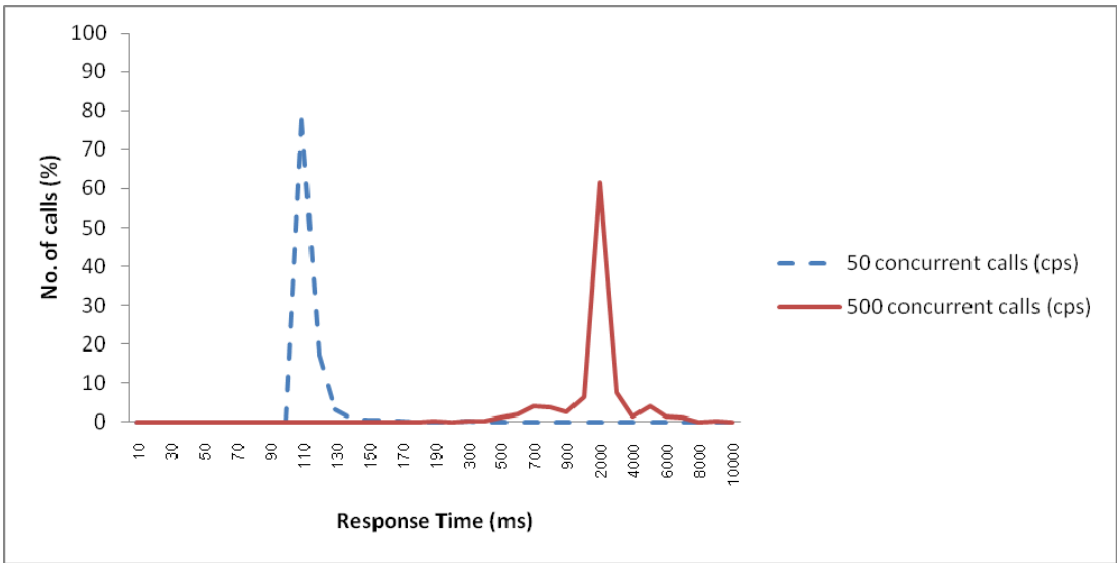


Figure 5.18: Compared Response Times of Proposed System

In addition, the response time between 0-10 ms of varied number of concurrent calls of the baseline system and of the proposed system are measured and plotted as graphs as shown in Figures 5.19 and 5.20, respectively. Obviously, for both systems, the response time falls after 100 concurrent calls. For the baseline system, it falls gracefully, but for the proposed system, it falls sharply.

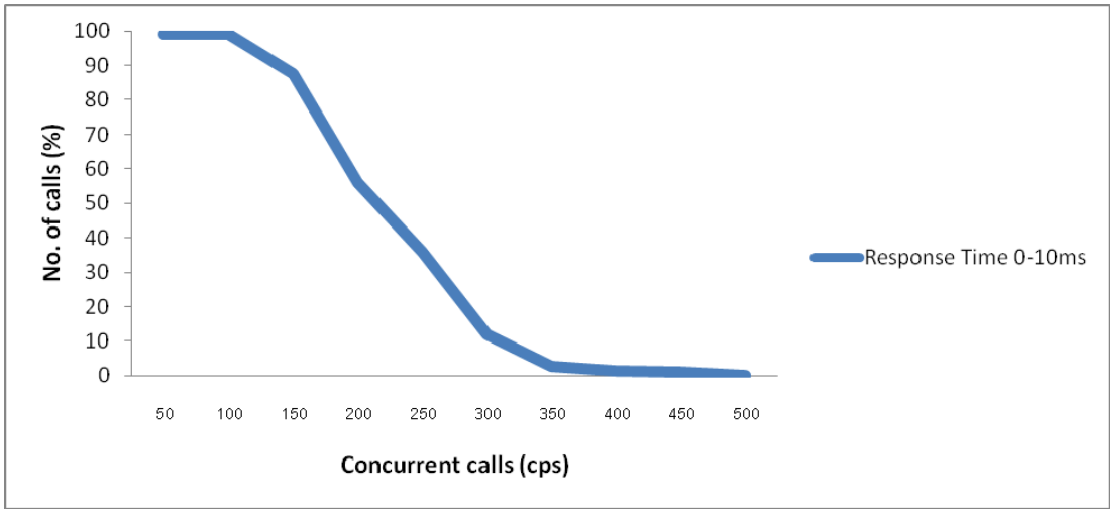


Figure 5.19: Response Time between 0-10 ms of Baseline System

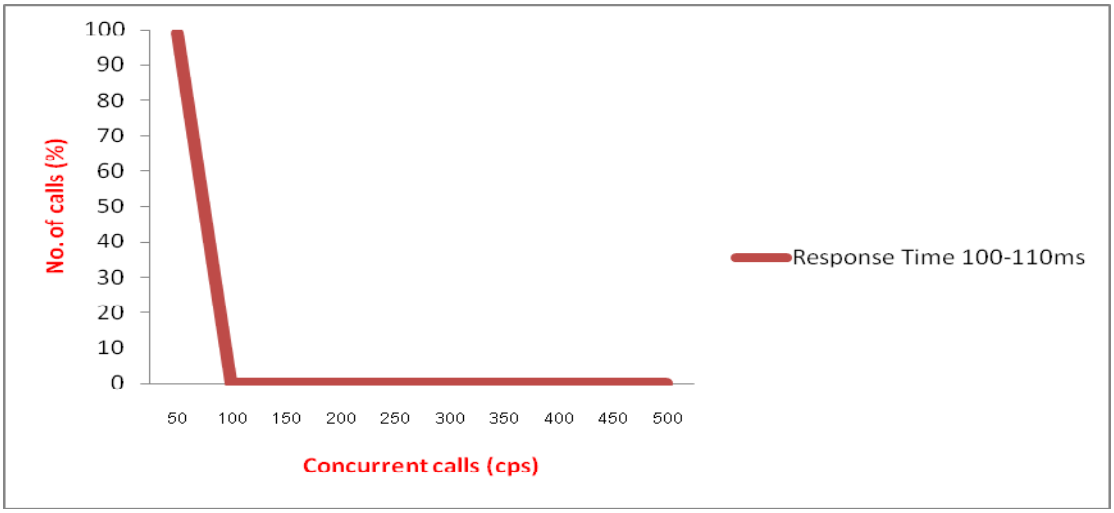


Figure 5.20: Response Time between 0-10ms of Proposed System

5.5.2 Failed Call Rate Analysis

Figures 5.21 and 5.22 illustrate the results of the failed call rates of the baseline system and the proposed system, respectively. The number of concurrent calls is varied between 50-150 calls. Clearly, in the baseline system, the failed call rates start to increase at 120 concurrent calls and increase slightly afterwards. However, in the proposed system, the failed call rates start to increase at 90 concurrent calls, and increase rapidly afterwards.

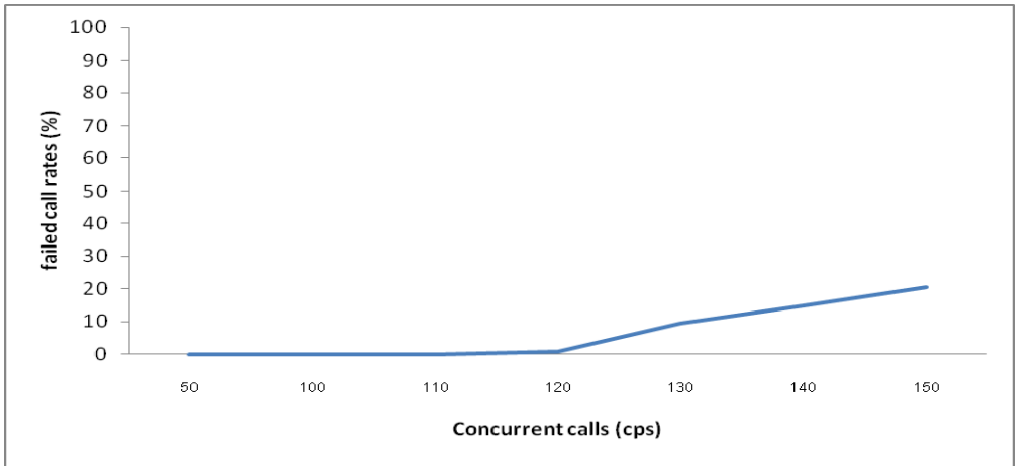


Figure 5.21: Failed Call Rates of Baseline System

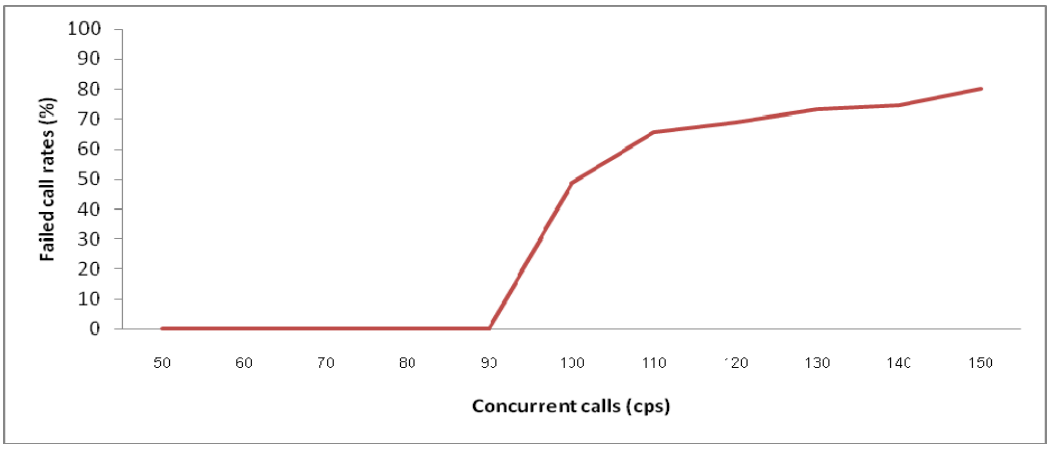


Figure 5.22: Failed Call Rates of Proposed System

5.5.3 Voicemail Call Analysis

Figures 5.23 show and compare the results of the response time obtained on the four test case conditions. The response time of normal voice messages, voice messages with AES encryption over the TCP and SSL are measured. Obviously, the response times of the four cases are different due to the encryption cost rather than the protocol used. Specifically, the response times when using TCP and SSL are not much different either with or without encryption. Encryption clearly affects the response times on retrieving voice messages due to more processing time and the increasing file size. In addition, the response time for encryption cases is almost doubled when the file size is high.

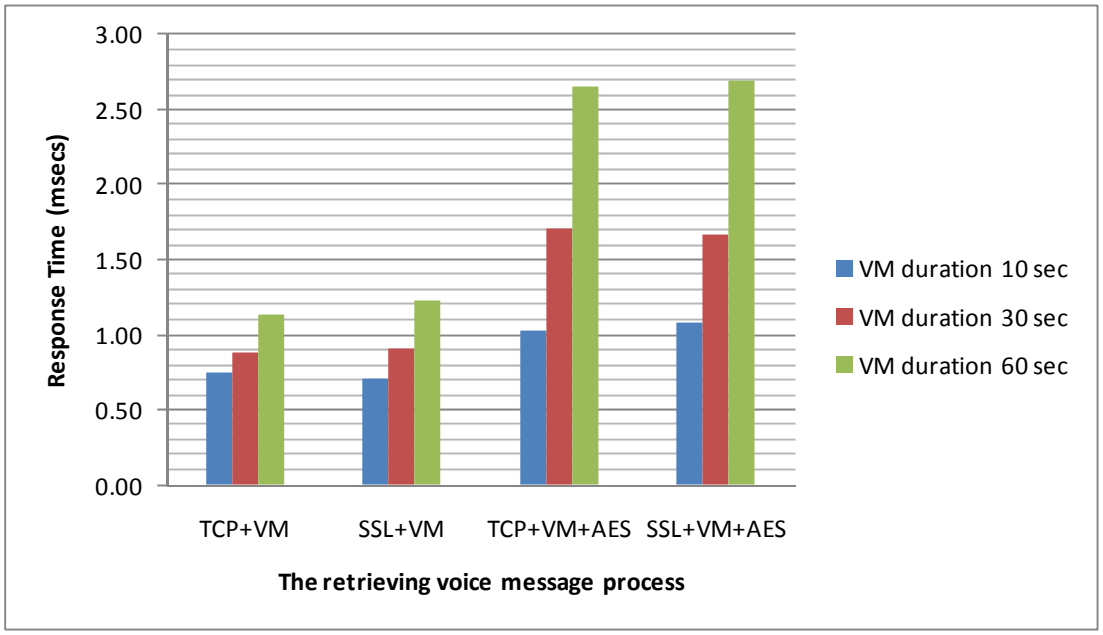


Figure 5.23: Response Times of Encrypted Voice Messages of Proposed System

5.6 Summary of Experimental Results

The response time of both systems clearly increases when having more concurrent calls but the failed call rates has slightly different results. Particularly, the response time in the baseline system is acceptable at 100 concurrent calls since it is increased drastically when the number of concurrent calls is increased. At 100 concurrent calls, the response time is reasonable between at 0-10 ms on the baseline system and between 100-110 ms on the proposed system. In addition, the failed call rate increases at 120 concurrent calls on the baseline system and at 90 concurrent calls on the proposed system. Thus, the effect of encryption is obvious.

The voicemail call has minimal effect to the response time when retrieving voice messages using the secure protocol, SSL. On the contrary, the encryption of voice messages for confidentiality incurs higher cost. Thus, adding encryption to the secure voicemail process should be considered with the high response time in mind.

In summary, these experiments indicate that the performance of the baseline system is acceptable at 120 concurrent calls and at 90 concurrent calls for the proposed system. Thus, the number of concurrent calls is decreased by approximately 25% when adding security features to the baseline system.

CHAPTER VI
DISCUSSION AND CONCLUSION

6.1 Summary of Our Contribution

Our work aims to measure the performance of the voice mail communication when adding the security to the baseline system as well as to provide some convenience for effective communication. An Asterisk server is functioned as a PBX for Internet telephony facilities. Even though the Asterisk server can provide and serve like an effective PBX services, it is not seriously concerned on the security issues. We enhance the security of the Asterisk server in three aspects. SIP signaling is transported on a secure channel via the TLS. Our work also encrypts voicemail messages and stores them in the database.

For security and management, some configurations on the Asterisk have to be modified. The main configuration files are the Asterisk *.conf files and these files need to be reloaded every time they are edited. The ARA or Asterisk Real time Architecture is designed to allow the mapping of the configuration files to the tables in the database. The advantage is that the configuration can be changed in real time and there is no need for reloading. Moreover, the voicemail is stored in a user's mailbox directory on the same server. It is inevitable that the storage for voicemail messages will always increase. Thus, the voicemail messages should be kept in the separate database from the voicemail database. In addition, the AES encryption is applied for protecting the voicemail messages before storing them into the database. The secure communication between the Asterisk server and the voicemail database server is done via the secure HTTP channel. system.

6.2 Problems and Limitations

Our research and experiments have some limitations. Firstly, it would be the high cost for testing in the real deployment. Thus, we run our experiments using

the testing tool for generating calls and self-defined activities. Later, the results are summarized and analyzed. The communication network used is not our concern, and the testing is done between a test computer and the server only. For real services, the behavior of users can be complicated and common usage patterns as the test scenarios may not be found. The unexpected errors may occur and cannot be reconstructed for testing. This work is designed for a fully VoIP system with voicemail services. It also requires support for connecting to PSTN for widely communication services. In addition, the Asterisk document notes that the TCP and TLS supports for SIP are currently considered as an experimental stage.

For the voicemail service testing, a set of experiment is run to measure the response time in retrieving voice messages. As the system just starts, we assume that the small volume of voice messages exists.

6.3 Conclusions and Future Work

This work shows the effect of adding security features to the performance of a voicemail system. We use the response time at different concurrent calls and the failed call rates as the performance metrics. The important testing tool used is SIPp which is the standard tool obtained from SPECSIP. The results illustrate the limitation of the Asterisk server.

Our implementation of the proposed system is based on Asterisk and FreeBSD operating system. The benchmark can also be tested on other operating systems such as Ubuntu, SUSE, or any Linux distribution. In addition, the performance of other PBX software using the SIP protocol including the open source software such as sipXecs can be investigated. Finally, the selected benchmark can be used to compare and evaluate many issues such as the performance, usability, scalability, PBX features, ease of management, and ease of use.

REFERENCES

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

[2] Lisa J. Stifelman, *Not Just Another Voice Mail System*, MIT Media Laboratory, Cambridge MA, 02139, Proceedings of 1991 Conference. America Voice I/O Society.

[3] Eduardo B. Fernandez, Juan C. Pelaez and Maria M. Larrondo-Petrie, "*Security patterns for Voice over IP Networks*", Journal of software, vol.2, no.2, August 2007.

[4] *AES Voice Encryption*, Available at http://venus.ece.ndsu.nodak.edu/

[5] Pauli Vesterinen, "*User authentication in SIP*", TKK T-110.5290 Seminar on Network Security,2006-12-11/12

[6] Aytunc Durlanik, and Ibrahim Sogukpinar, "*SIP Authentication Scheme using ECDH*", PWASET Volume 8 October 2005 ISSN 1307-6884.

[7] JoongMan Kim, SeokUng Yoon, HyunCheol Jeong, YooJae Won, "*Implementation and Evaluation of SIP-based Secure VoIP Communication System*", IT Infrastructure Protection Division, Applied Security Technology Team, Korea Information Security Agency, Seoul, Korea, 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing.

[8] Razvan Rughinis and Cristian Iconaru, "*A Practical Analysis of Asterisk SIP Server Performance*", Politehnica University of Bucharest Romania and RoEduNet Romania, the 7-th International Conference RoEduNet 2008 in Cluj-Napoca, Romania.

[9] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "*RTP: A Transport Protocol for Real-Time Applications*", RFC 1889, Internet Engineering Task Force, 1996.

[10] M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman, "*The Secure Real-time Transport Protocol (SRTP)*", RFC 3711, Internet Engineering Task Force, 2004.

[11] P. Zimmermann, J. Callas, "*ZRTP: Media Path Key Agreement for Secure RTP*", draft-zimmermann-avt-zrtp-06, 2008.

[12] ITU-T Recommendation H.323, Available at http://en.wikipedia.org/wiki/H.323

[13] Transport Layer Security(TLS), Available at http://en.wikipedia.org/wiki/Transport_Layer_Security

[14] Advanced Encryption Standard (AES), Available at http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[15] E-model ITU-T Recommendation G.107 (03/2005), Available at http://www.itu.int/ITU-T/studygroups/com12/emodelv1/index.htm

[16] Assessing VoIP Call Quality Using the E-model by John Q. Walker NetIQ Corporation, Available at http://www.ixiacom.com/library/white_papers/display?skey=voip_quality

[17] PROTOS- Security Testing of Protocol Implementations, Available at http://www.ee.oulu.fi/research/ouspg/protos/

[18] Voice over Internet Protocol (VoIP), Available at http://en.wikipedia.org/wiki/Voice_over_Internet_Protocol

[19] IP (Internet Protocol) PBX (Private branch exchange), Available at http://en.wikipedia.org/wiki/IP_PBX

[20] The Open Source PBX, Available at http://www.asterisk.org/

[21] Asterisk, Available at http://www.voip-info.org/wiki/view/Asterisk

[22] Voicemail, Available at http://en.wikipedia.org/wiki/Voicemail

[23] Inter-Asterisk eXchange protocol, Available at http://en.wikipedia.org/wiki/Inter-Asterisk_eXchange

[24] SIPp, Available at http://sipp.sourceforge.net/

[25] CODEC, Available at http://en.wikipedia.org/wiki/Codec

[26] OpenSSL, Available at http://www.openssl.org/

[27] The FreeBSD Operating System, Available at http://www.freebsd.org/

[28] Comparison of VoIP software, Available at http://en.wikipedia.org/wiki/Comparison_of_VoIP_software

[29] กิตติพงษ์ สุวรรณราช, "*ออกแบบ และติดตั้งระบบโทรศัพท์ IP-PBX ด้วย Asterisk*", กรุงเทพฯ : ออฟ เซ็ท เพรส, กุมภาพันธ์ 2551

[30] Standard Performance Evaluation Corporation, SPECSIP, Available at http://www.spec.org/specsip/

APPENDICES

APPENDIX A
INSTALLATION GUIDELINE

A.1 Introduction

In this research project, there are 2 servers such Asterisk Server and Voicemail Database Server which both runs on FreeBSD 8.0. The necessary packages are offered by FreeBSD Ports and Packages Collection to simply install applications.

A.2 Requirements

It lists the applications and the necessary packages which must install for each server after OS preparation. The application name and version number are referred by the experiment in this work. In addition, it simply has to prepare download these packages in "/usr/ports/distfies/" directory.

- Asterisk Server
  - Apache
    - httpd-2.2.13.tar.bz2
  - MySQL
    - mysql-5.0.86.tar.gz
  - PHP
    - php-5.2.11.tar.bz2
    - suhosin-patch-5.2.11-0.9.7.patch.gz
    - php-5.2.10-mail-header.patch
  - Asterisk and Asterisk-addons
    - asterisk-1.6.0.15.tar.gz
    - asterisk-addons-1.4.6.tar.gz
  - OpenSSL
    - openssl- 0.9.8k.tar.gz
    - dtls-bugs-2009-05-18.patch

- Voicemail Database Server
  - Apache
    - httpd-2.2.13.tar.bz2
  - MySQL
    - mysql-5.0.86.tar.gz
  - PHP
    - php-5.2.11.tar.bz2
    - suhosin-patch-5.2.11-0.9.7.patch.gz
    - php-5.2.10-mail-header.patch
  - OpenSSL
    - openssl- 0.9.8k.tar.gz
    - dtls-bugs-2009-05-18.patch
  - SIPp
    - sip.2009-01-21.tar.gz

A.3 Packages Installation

For each application, the steps and commands are shown as below

(1) Apache

cd /usr/ports/www/apache22/

- To enable SSL, MySQL support and select these options by this command

make config

make install clean

- For automatically start this serviec every time boot, it must to add this ‘apache22_enable= “YES” ’ to /etc/rc.conf.
- The configuration file, httpd.conf is at /usr/local/etc/apache22/
- The Document root assigns default as /usr/local/www/apache22/data/

(2) MySQL

cd /usr/ports/databases/mysql50-server/

make WITH_CHARSET= tis620

WITH_XCHARSET= all
WITH_COLLATION= tis620_thai_ci
WITH_OPENSSL= yes
WITH_NDB= yes
WITH_INNODB= yes
WITH_MYISAM= yes
WITH_FEDERATED= yes
WITH_ARCHIVE= yes
WITH_OPTIMIZED= yes install clean

- To define password for root user by this command

mysqladmin -u root password *new password*

- For automatically start this serviec every time boot, it must to add this ‘mysql_enable= “YES” ’ to /etc/rc.conf.

(3) PHP

cd /usr/ports/lang/php5/

- To enable Apache, SSL support and select these options by this command

make config
make install clean
cd /usr/ports/lang/php5-extensions/
make install clean

- To assign configuration file as php.ini by this command

cp /usr/local/etc/php.ini-dist /usr/local/etc/php.ini

(4) Asterisk

cd /usr/ports/net/asterisk16/

- To enable SSL, MySQL and select these options support by this command

make config
make install clean
cd /usr/ports/net/asterisk16-addons/
make install clean

- For automatically start this serviec every time boot, it must to add this ‘asterisk_enable= “YES” ’ to /etc/rc.conf.
- The configuration files are at /usr/local/etc/asterisk/

(5) OpenSSL

cd /usr/ports/security/openssl/

make install clean

(6) SIPp

cd /usr/ports/benchmarks/sipp/

- To enable SSL support and select this option by this command

make config

make install clean

APPENDIX B
SCENARIO

B.1 Completed call scenario

The scenario is shown as XML file. It is run by sipp command to generate calls. A name of completed call scenario is defined as “invite.xml” as shown below.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">
<!--              Sipp default 'uac' scenario.       -->
<scenario name="call scenario">
  <!-- In client mode (sipp placing calls), the Call-ID MUST be       -->
  <!-- generated by sipp. To do so, use [call_id] keyword.             -->
  <send retrans="500">
    <![CDATA[
      INVITE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[pid]SIPpTag00[call_number]
      To: sut <sip:[service]@[remote_ip]:[remote_port]>
      Call-ID: [call_id]
      CSeq: 1 INVITE
      Contact: sip:sipp@[local_ip]:[local_port]
      Max-Forwards: 70
      Subject: Performance Test
      Content-Type: application/sdp
     Content-Length: [len]
      v=0
      o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
```

```
    s=-
    c=IN IP[media_ip_type] [media_ip]
    t=0 0
    m=audio [media_port] RTP/AVP 0
    a=rtpmap:0 PCMU/8000
   ]]>
 </send>
 <recv response="100" optional="true">
 </recv>
 <recv response="180" optional="true">
</recv>
 <recv response="183" optional="true">
</recv>
 <!-- By adding rrs="true" (Record Route Sets), the route sets         -->
 <!-- are saved and used for following messages sent. Useful to test   -->
 <!-- against stateful SIP proxies/B2BUAs.                             -->
 <recv response="200" rtd="true" rrs="true">
</recv>
 <!-- Packet lost can be simulated in any send/recv message by         -->
 <!-- by adding the 'lost = "10"'. Value can be [1-100] percent.       -->
 <send>
  <![CDATA[
    ACK [next_url] SIP/2.0
    Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
    From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[pid]SIPpTag00[call_number]
    To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
    Call-ID: [call_id]
    CSeq: 1 ACK
    Contact: sip:sipp@[local_ip]:[local_port]
    Max-Forwards: 70
    Subject: Performance Test
    Content-Length: 0
```

```
    ]]>
  </send>
  <!-- This delay can be customized by the -d command-line option       -->
  <!-- or by adding a 'milliseconds = "value"' option here.            -->
  <pause/>
  <!-- The 'crlf' option inserts a blank line in the statistics report. -->
  <send>
    <![CDATA[
      BYE [next_url] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[pid]SIPpTag00[call_number]
      To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
      Call-ID: [call_id]
      CSeq: 2 BYE
      Contact: sip:sipp@[local_ip]:[local_port]
      Max-Forwards: 70
      Subject: Performance Test
      Content-Length: 0
    ]]>
  </send>
  <recv response="200" crlf="true">
 </recv>
<!-- definition of the response time repartition table (unit is ms)   -->
  <ResponseTimeRepartition value="10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120,
130, 140, 150, 160, 170, 180, 190, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2000,
3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000"/>
  <!-- definition of the call length repartition table (unit is ms)     -->
  <CallLengthRepartition value="100, 200, 300, 4000, 500, 600, 700, 800, 900,
1000"/>
</scenario>
```

BIOGRAPHY

NAME	Mr.Chalearmpop Ongsit
DATE OF BIRTH	11 June 1972
PLACE OF BIRTH	Chantaburi, Thailand
INSTITUTIONS ATTENDED	Ramkhamhaeng University, 1998 : Bachelor of Science (Computer Science) Mahidol University, 2010 : Master of Science (Computer Science)
POSITION&OFFICE	2001-Present, Revenue Department of Thailand, Bangkok, Thailand Position : Computer Technical Officer Tel. 0-2272-8714 E-mail: chalearmpop.on@rd.go.th
ADDRESS	The Revenue Department, 90 Soi Phaholyothin7, Phaholyothin Road, Bangkok 10400, Thailand