Java Implementation of Smart Permit Architecture for a Secure Wireless VoIP Enterprise network with Asterisk and SIP Communicator Integration

Main Project Report

By
Divya Krishnan
Parvathy S
R Vidya Lakshmi
Vishnudatha K

Under the guidance of

Br. Jayaraj Poroor

Department of Computer Science and Engineering,
Amrita School of Engineering
Amritapuri,
Kollam-690525
2008-2009

Java Implementation of Smart Permit Architecture for a Secure Wireless VoIP Enterprise network with Asterisk and SIP Communicator Integration

Main Project Report

By
Divya Krishnan
Parvathy S
R Vidya Lakshmi
Vishnudatha K

Under the guidance of

Br. Jayaraj Poroor

श्रद्धावान् लभते ज्ञानम्

Department of Computer Science and Engineering,
Amrita School of Engineering
Amritapuri,
Kollam-690525
2008-2009

AMRITA SCHOOL OF ENGINEERING

Managed by: Mata Amritanandamayi Math

Department of Computer Science

Aum Amriteswaryai Namah

CERTIFICATE

*Certified that this report entitled "**Java Implementation of Smart Permit Architecture for a secure wireless VoIP enterprise network with Asterisk and SIP communicator integration**" is a bonafide report of the main project submitted by Divya Krishnan, Parvathy S, R Vidya Lakshmi, Vishnudatha K in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science & Engineering of Amrita Viswa Vidyapeetham during the year 2009.*

Project Guide

Mr. Jayaraj Poroor

CTO, ARL

Place: Amritapuri
Date: 30.04.2009

Amritapuri, Vallikkavu, Clappana P. O., Kollam-690525
Ph: 0476-2898329, 2829318, 2898328
2896318, 2896328, Fax: 0476-2896178, Email: aitec@amritapuri.amrita.edu. Website: aitec.amrita.edu/amritapuri
Branches: ● Ettimadai ● Bangalore

ACKNOWLEDGEMENT

We offer our humble pranams at the lotus feet of Sri Satguru MATA AMRITANANDAMAYI DEVI, our spiritual mother who is the guiding light of our lives. This project has been carried out as a part of our B Tech Curriculum and it is submitted with extreme gratitude and pleasure.

We make use of this opportunity to express our sincere gratitude to all those who have helped us to complete this project successfully. We would like to convey our sincere thanks to Dr. K. SANKARAN, principal, ASE Amritapuri, and Head of our Department, Mr. M KRISHNAKUMAR for their help.

We are extremely thankful to our project guide, Br. JAYARAJ POROOR CTO Amrita Research Labs, whose guidance and support has gone a long way to our successful completion of our project. We extend our sincere thanks to Mr Anish Chandran, our Department guide, Mr Anoop, Manager ICTS, Br. Sundar Gopalan, Br. Karthi , Mr Shinit , Mr Vipin Pavithran, ARL staff and all the staff of ASE for providing the valuable help necessary for the completion of the project.

We would also like to express our sincere thanks to our friends and classmates for their encouragement.

ABSTRACT

The project is aimed at the implementation of Smart Permit Architecture for a secure wireless VoIP enterprise network using Java with Asterisk and SIP Communicator integration. The Smart Permit Architecture is a light weight architecture for permitting guest-devices into Secure 802.11 Enterprise Wireless LANs. The device authentication is performed using the Device Whisper protocol which uses the concept of audio based location limited channel. The system developed enables a SIP communicator to enter into secure communication with Asterisk server by establishing a secure channel.

CONTENTS

1. INTRODUCTION

Voice over Internet Protocol (VoIP) is becoming extensively used in communication nowadays due to its reduced cost and availability. It significantly reduces the per minute cost, resulting in reduced long-distance bills. Thus VoIP services are becoming more and more prevalent in enterprises.

1.1 NEED FOR THE SYSTEM

As VoIP services are becoming more and more prevalent these days, the need to enable a wireless guest device to securely register with an enterprise VoIP network arises. In order to enable the authentication of the guest device, the system developed should be light weight and involve minimum human intervention.

1.2 FEATURES OF THE SYSTEM

- Should be light weight that is it shouldn't impose any cross organizational infrastructure to be setup and also should minimize the passing of credentials
- Should be platform independent that is guest device running on any operating system and with any hardware specifications should be able to use this system
- Should be compatible with Asterisk server
- Should be compatible for any hardware
- Should be compatible with SIP Communicator

1.3 OVERVIEW OF THE SMART PERMIT ARCHITECTURE AND RELATED CONCEPTS

The project is aimed at the Java implementation of the Smart Permit architecture for a secure wireless VoIP enterprise network with Asterisk and SIP Communicator integration.

The smart permit architecture introduces a lightweight architecture for permitting guest-devices into Secure 802.11 Enterprise Wireless LANs. Enterprise Wireless LANs consist of an Access Control Server which is responsible for providing and managing the access credentials to devices to access the network. The architecture introduces the concept of a trust bootstrap gateway(TBG) which has a trusted secure relationship with the Access Control Server. The TBG is which is placed at a location which is accessible only to privileged guests.The TBG provides an insecure wireless link which enables guest devices can connect to it. An applet signed by a root certification authority is loaded from the TBG to the client device. As soon as the applet is loaded, a self signed certificate of the client is generated. A TLS connection is established between the device and the TBG where the certificate of the server and client are exchanged. The client is able to validate the server by comparing the certificate obtained through TLS handshake and that obtained through the applet. It is the TBG which is responsible for verifying the identity of the guest device and

hence for establish trust bootstrapping between the guest device and the ACS. The process of authenticating the guest device is performed through Device Whisper protocol which is based on audio based location limited channel. The guest device generates a set of audio sampled based on the cryptographic hash of the guest device's self signed certificate. This is recorded by the TBG and is decoded to produce the original hash. This hash is compared with the hash of the client certificate obtained after TLS handshake and hence the client is verified. The location-limited audio channel ensures that only privileged guests are able to enter the network. On successful verification of the client by the TBG, a trusted channel is established between the ACS and the guest device whereby information can be exchanged between them and the guest can avail the services of the network.

1.4 GOALS AND OBJECTIVES

Implementing Smart Permit Architecture using java for a secure wireless VoIP enterprise network with Asterisk PBX and SIP Communicator integration.

- Java implementation of Trust Bootstrap Gateway with Asterisk integration
- Implementation of Smart permit Client device using Java Applet with Sip Communicator integration
- Device Whisper Protocol implementation using JSSE and Java Sound API

2. LITERATURE SURVEY

Literature survey is a detailed study of the various operations performed by a system. This involves gathering information and using structured tools for analysis. Once the analysis is completed, the analyst has a firm understanding of what is to be done. The next step is to decide how the problem might be solved. So system analysis and design are the application system approaches to problem solving, generally using computers. To reconstruct a system, the analyst must consider its elements outputs and inputs processors, controls, feedback and environment.

2.1 EXISTING SYSTEMS

The analysis part includes a detailed study of the existing system. Many limitations were identified on the existing system. All these in one way or the other harm the effective functioning of the overall processes.

- EDUROAM - used by traveling researchers to securely access WLANs when visiting participating organizations. This system is based on transcontinental infrastructure and requires that both the visitor's institution and the visited institution be part of this infrastructure.

- Greenpass System – This system uses current wireless security frameworks to allow guest access to a secured wireless network and selected internal resources employ a human-assisted scheme based on visual hashes for bootstrapping trust relationship between guest-device and an authorized user's device.

- Network-in-a-box - uses an enrollment station implementing infrared-based location limited channel for bootstrapping trust and issuing WLAN authorization certificate to fresh devices.

2.2 DRAWBACKS OF THE EXISTING SYSTEMS

All the existing systems mentioned above had their limitations. So the proposed system was designed to overcome these limitations.

- EDUROAM – It requires the participating organizations to have transcontinental infrastructure between the organizations.

- Greenpass System – Even though, it is a light weight architecture it employs human assistance for establishing trust bootstrapping.

- Network-in-a-box – As it uses infrared-based location limited channel for bootstrapping trust and issuing WLAN authorization, it may not be widespread in all devices.

2.3 PROPOSED SYSTEM

The proposed system is the one, which is to be developed which will clear the drawbacks of the existing system. The proposed system is the implementation of smart permit architecture. Smart permit architecture is a lightweight architecture for permitting guest-devices into Secure Enterprise Wireless LANs. It consists of a Trust Bootstrap Gateway to establish a trusted secure channel between the guest device and Access Control Server. TBG has a long term trust relationship with the ACS.TBG provides an insecure wireless link which enables fresh devices to get connected with the gateway. It introduces the Device Whisper protocol for authenticating the guest device which uses the concept of audio based location limited channel.

The various advantages of proposed system are as follows:

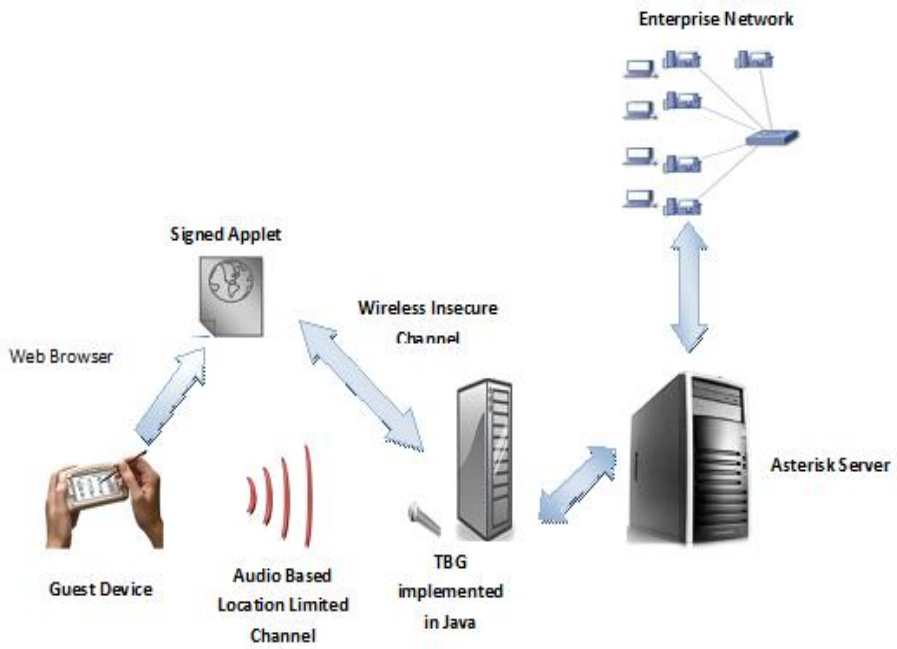
- The system is easy to use and it reduces human intervention.
- The system is light weight and is platform independent.
- The system uses audio based location limited channel which is available in all devices that are to be authenticated.
- Location limited channel ensures that the insecure wireless network around the TBG is not exploited by an attacker.
- It ensures secrecy as the data is passed as hashes over the location limited channel.
- Location-limited channels have the property that human operators can precisely control which devices are communicating with each other.

Main modules:

1. Java implementation of Trust Bootstrap Gateway with Asterisk integration
2. Implementation of Smart permit Client device using Java Applet with Sip Communicator integration
3. Device Whisper Protocol implementation using JSSE and Java Sound API

3. SYSTEM DESIGN

3.1 ARCHITECTURE DIAGRAM:

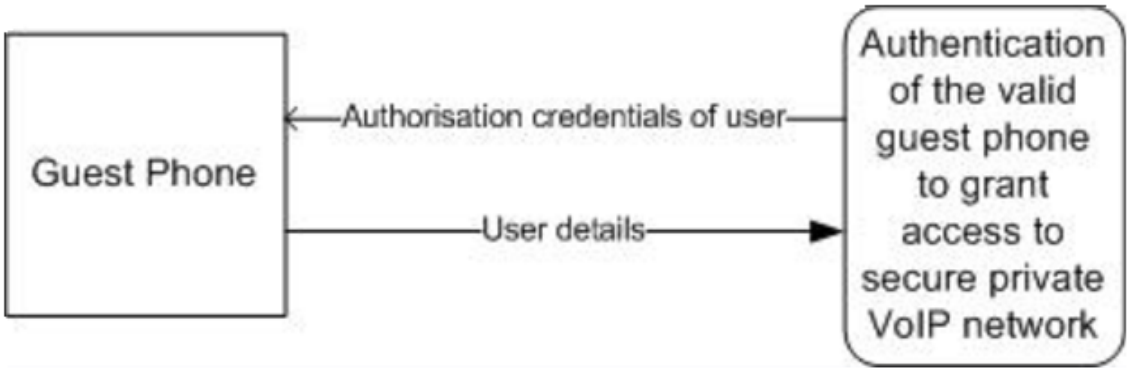


The system allows the secure entry of a guest phone into a secure VoIP enterprise network using audio-based location-limited channel. The system consists of a Trust Bootstrap gateway which is placed at a location which is accessible only to privileged guests. The guest is required to bring the guest phone near the gateway. The device credentials are passed on to the gateway via the insecure wireless network established with the Trust Bootstrap gateway. Meanwhile the credentials are also passed via audio-based location-limited channel. The credentials obtained via both the channels are verified by the gateway for integrity. When the credentials are being passed through the audio-based channel, the environment should be noise-free. The location-limited audio channel ensures that only privileged guests are able to enter the network. On successful verification, the guest device is registered with the Asterisk server and is given an extension number which is loaded on to the sip communicator in the client device.

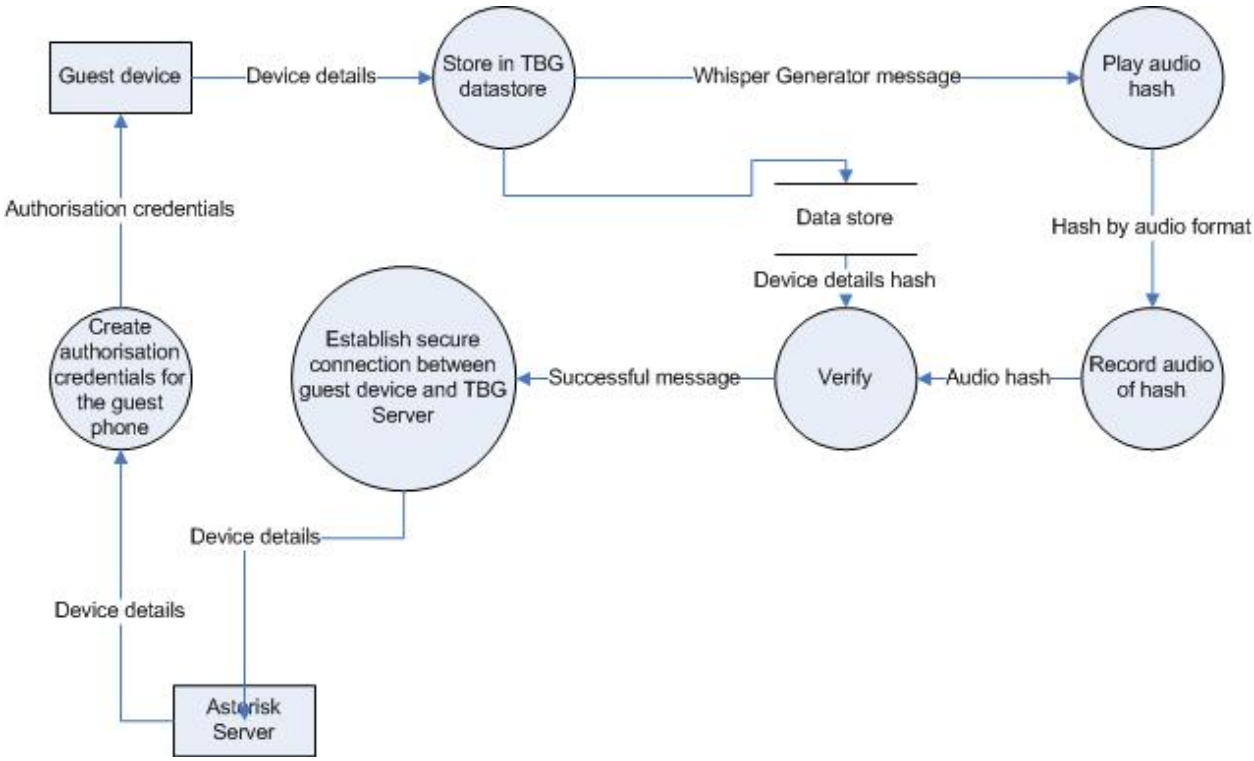
3.2

DATA FLOW DIAGRAM

DFD Level 0 Diagram



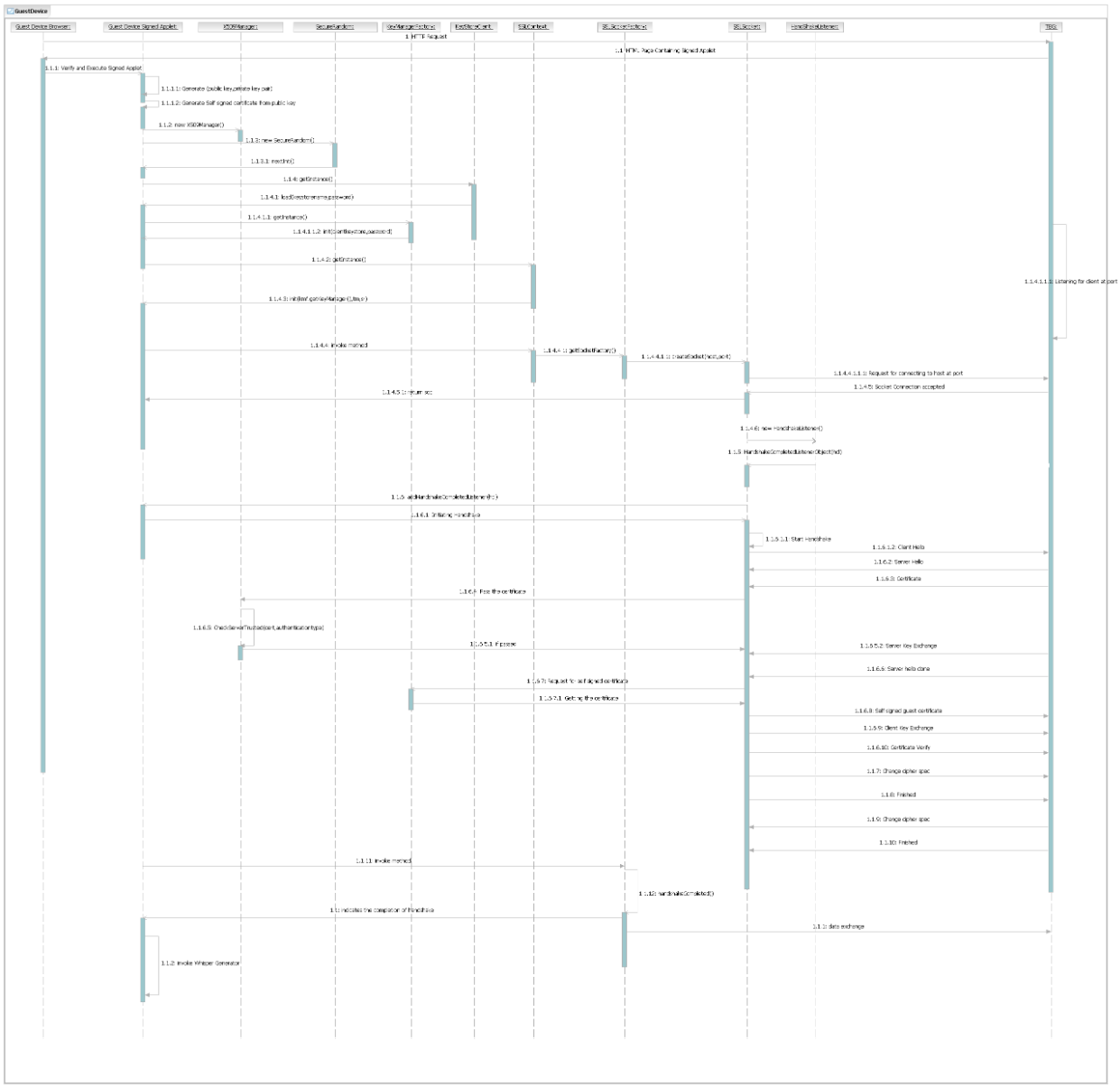
DFD Level 1 Diagram



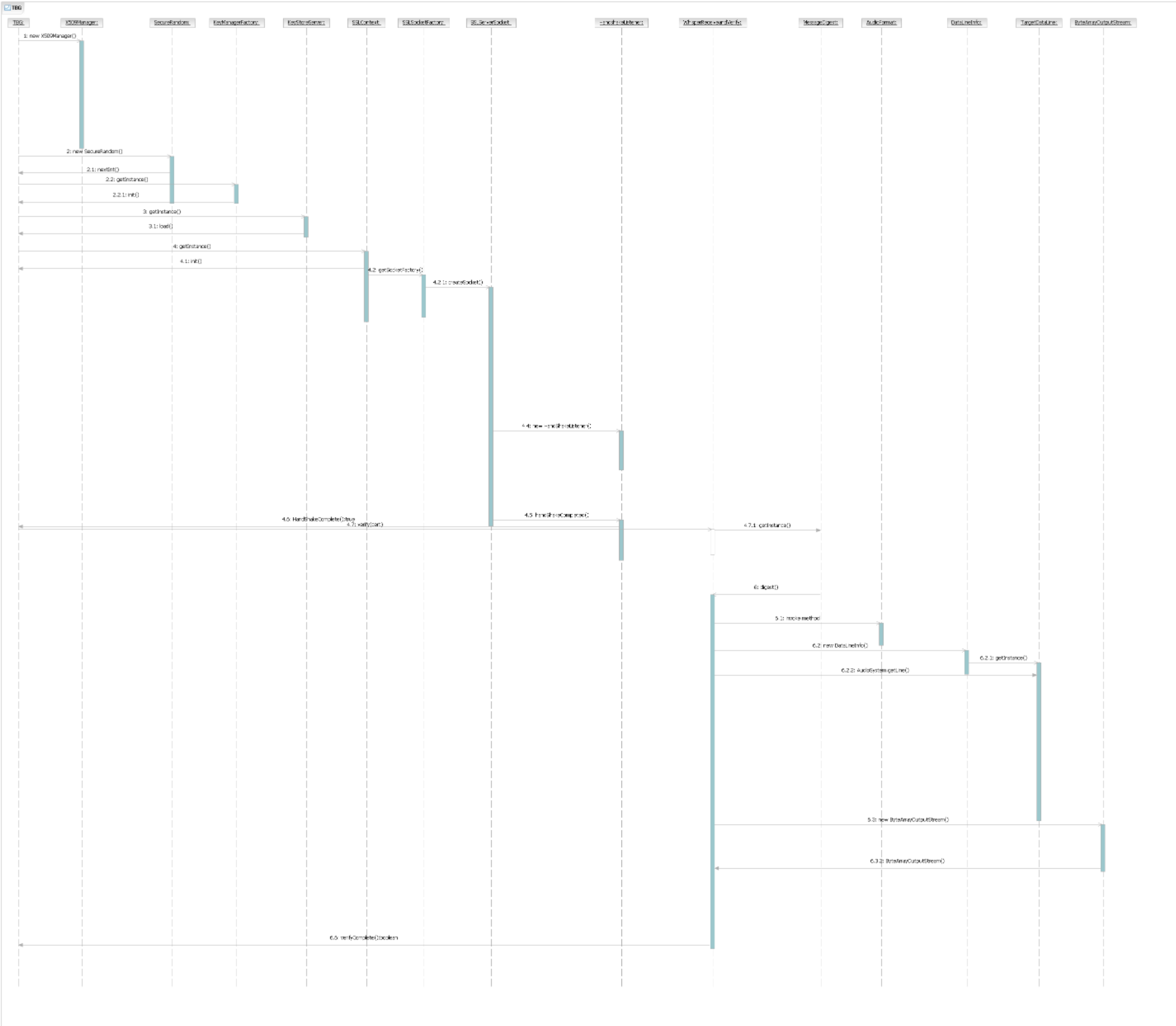
3.3

SEQUENCE DIAGRAM

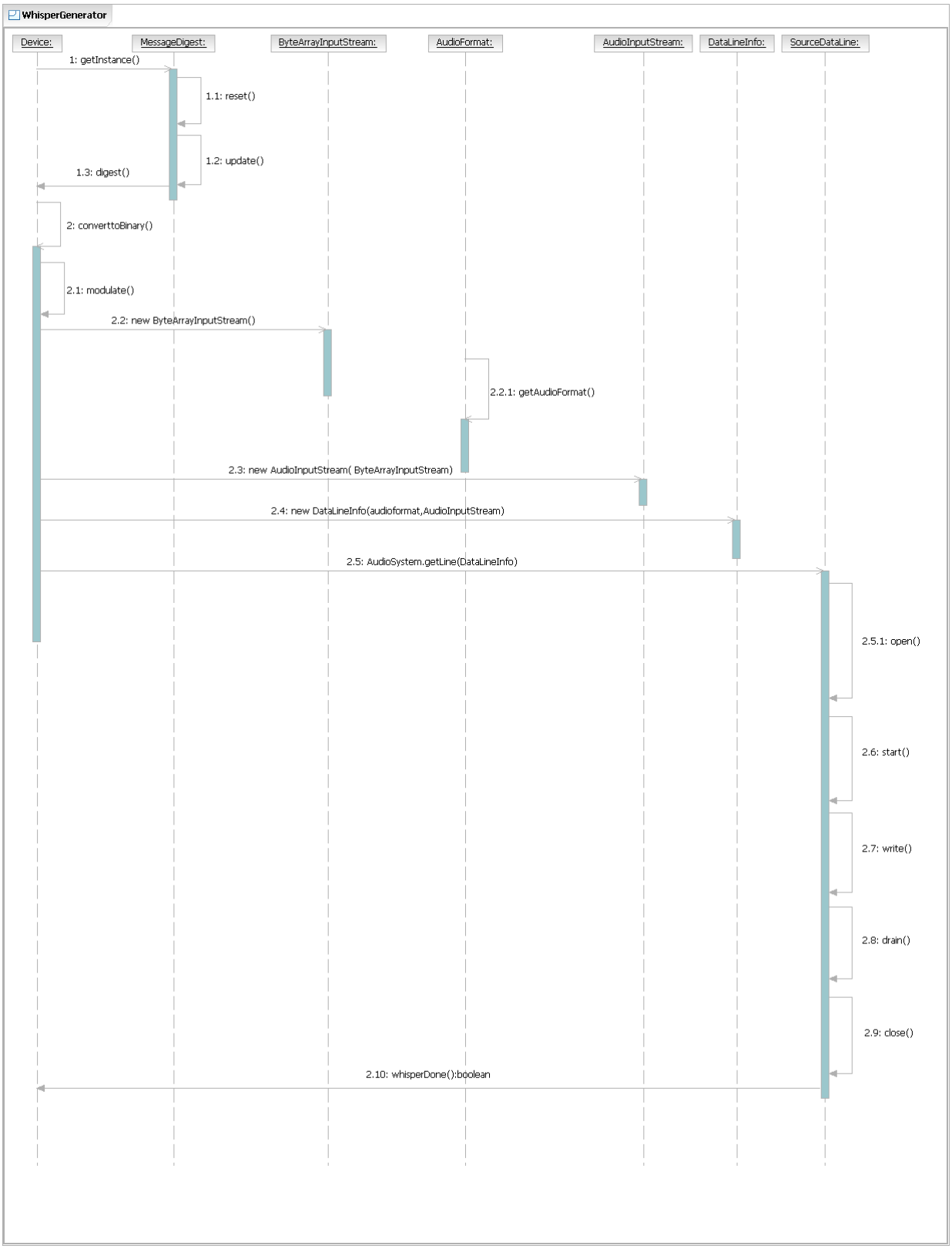
Client



TBG Server



Whisper Generator



4. OBSERVATIONS

Device Whisper Protocol is implemented in matlab. The implementation was tested real-time with different hashes. The variations in the recorded samples in different formats were observed. The influence of noise in the recorded signals was observed and a processing algorithm is designed according to these observations. We implemented the smart permit architecture in Java based on these observations.

5. METHODOLOGY

5.1 SOFTWARE ENGINEERING PARADIGM

WATERFALL MODEL

The waterfall model is a sequential software development model (a process for the creation of software) in which development is seen as flowing steadily downwards (like a waterfall) through the phases of requirements analysis, design, implementation, testing (validation), integration, and maintenance. The origin of the term "waterfall" is often cited to be an article published in 1970 by Winston W.Royce (1929-1995), [1] although Royce did not use the term "waterfall" in this article. Ironically, Royce was actually presenting this model as an example of a flawed, non-working model. (Royce 1970)

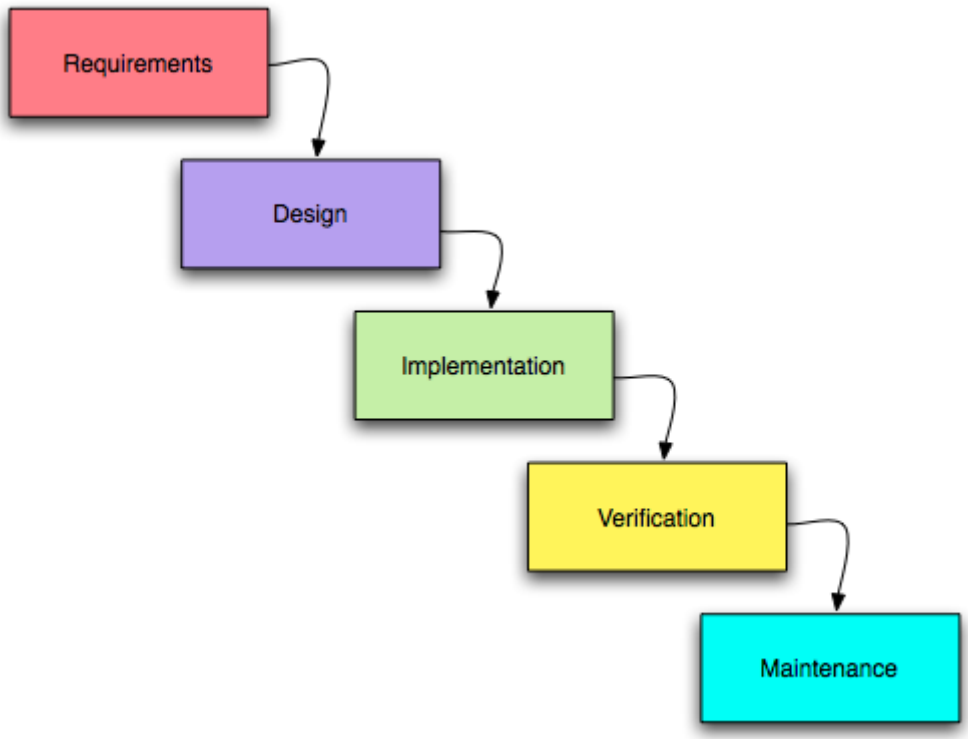
In Royce's original waterfall model, the following phases are followed in order: -

- Requirements specification
- Design
- Development
- Integration
- Testing and debugging
- Installation
- Maintenance

To follow the waterfall model, one proceeds from one phase to the next in a purely sequential manner. For example, one first completes requirements specifications, which are set in stone. When the requirements are fully completed, one proceeds to design. The software in question is designed and a blueprint is drawn for implementers (designers / programmers) to follow — this design should be a plan for implementing the requirements given. When the design is fully completed, an implementation of that design is made by designers / programmers. Towards the later stages of this implementation phase, distinct software components produced are combined to introduce new functionality and remove bugs.

Thus the waterfall model maintains that one should move to a phase only when it's preceding phase is completed and perfected. However, there are various modified waterfall models that may include slight or major variations upon this process.

However, there are various modified waterfall models (including Royce's final model) that may include slight or major variations upon this process.



6. IMPLEMENTATION AND TESTING

6.1 IMPLEMENTATION

The system consists of a trust bootstrap gateway implemented in Java which is placed at locations accessible by guests. The system requires that the enterprise has a certificate signed by any root Certification Authoritity. The guest is provided with a URL through which an applet which is signed by the root CA gets loaded in the client device. Now the applet starts running in the client.

A self signed certificate is created for the client after which a TLS connection is established by the client with the server. This is implemented using JSSE(Java Secure Socket Extension) package. A TLS(Transport Layer Security) connection establishes a secure channel between the peer entities. During the establishment of TLS connection, TLS handshake which is a series of ten steps, is done wherein the client and the server exchanges their certificates. The completion of TLS handshake results in the establishment of a secure channel between the client and the server through which data can be transferred securely. The client is able to verify the identity of the server by comparing the certificate obtained through the applet with the certificate obtained through TLS connection. Now the TBG has to verify whether the device which has loaded the applet and established TLS connection is privileged to access the network. This is done using Device Whisper Protocol which is based on audio based location limited channel.

We have implemented a location-limited channel (which we call DEVICEWHISPER channel) using a modulated 16 KHz carrier wave transmitted from speaker output of the guest device to microphone input of the TBG. The guest-device first generates a set of audio sample values based on the cryptographic hash of the guest-device's self-signed digital certificate, by executing DEVICEWHISPER-GENERATE algorithm.

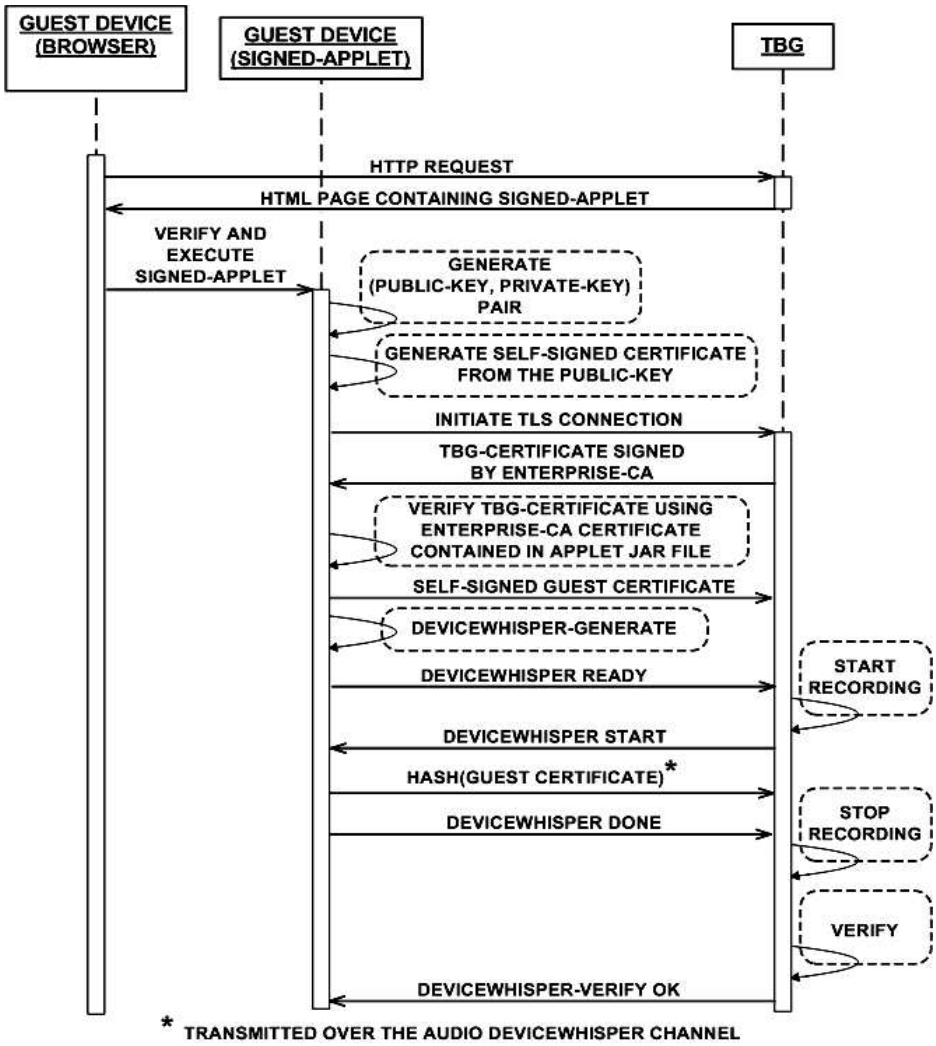
The guest-device plays the generated audio sample data via its speaker-output while the gateway records the audio. The TBG then executes DEVICEWHISPER-VERIFY algorithm, which demodulates and recovers the encoded bits from the recorded Audio samples. This hash is compared with hash of the certificate received during the TLS handshake. This enables the TBG to verify whether the TLS client-certificate it has received belongs to the near-by guest-device.

In device whisper generate we used SHA-512 algorithm for getting the hash of the certificate.

Then error correction bits are added to the generated hash. We used Hamming code as error correction mechanism. Then it performed Manchester Encoding on the hash bits and is modulated on 16KHz carrier wave using Amplitude-Shift-Keying. The audio samples are played using Java Sound API classes.

In the TBG the generated sound is recorded and processed using device whisper verify algorithm. Firstly the audio signal is recorded in time domain and it has to be converted to frequency domain. This is implemented by finding the fourier transform. We did it with windowing for increasing the accuracy. In order to cut all unwanted data from the recorded sample we use filter. The frequency domain data set is passed through a butterworth

filter for getting the noise free data. Then it is demodulated and the encoded bits are recovered from the audio samples.



The recovered bits are then compared with the certificates obtained through TLS handshake and is verified. If both the certificates are same the device will get authenticated.

On successful authentication an extension number is given to the client device which is registered with the asterisk PBX. On our implementation the extension number and a random password genrated from the asterisk server is written into the sip communicator that is installed in the client device. The password in the sip communicator.xml file is base 64 encodd and stored. Through this sip communicator installed within the client we can now place calls with the other clients registered with asterisk.

6.2 CONCEPTS

Signed Applet

An applet that includes a digital signature from its author that guarantees its source and integrity. The digital signature indicates who developed it and when, and whether it has been tampered with since that time. It allows users to verify the code being downloaded by their web browser has not been replaced during the transmission from server to client by a malicious attacker.

Usually an applet is signed using the certificate of the enterprise, obtained from a Root Certification Authority. When the signed applet is loaded into the browser, the browser traces the certificate of the applet to the authority who signed the certificate. Then it verifies that the certificate of the authority is already listed in the browser or not. If it is listed, it is taken that the applet is coming from a trusted source else the user is asked whether he/she will allow the applet to be trusted.

Advantages of Signed Applet

- Your signature is your guarantee to others that the applet is friendly.
- This adds accountability without the installation hassle or exposing your applet to attacks from outsiders.
- The signature does not prevent damage, but it serves as a stamp of good intention, and it identifies the responsible party if damage does occur.

Browser Security Architecture

- On Java virtual machines, by default, applications are implicitly trusted.
- The designers of the JVM specification assumed that users start applications at their own initiative; therefore, take responsibility for the application's behavior on their machine.
  Such code is considered to be trusted.
- However, applets are started automatically by the browser after it downloads and displays a page. Users cannot be expected to know what applets might contain before they download it; therefore, cannot take responsibility for the applet's behavior on their machine.
  Applets are considered by default to be untrusted.

After the applet is signed, the next time you or someone else downloads it in its page the browser will present a dialog box displaying the credentials you just created for it and asking the user permission to run it. If he/she chooses not to, the applet will throw the AccessControlException in the Java Console window in our browser. The difference is that now the user gets to make an informed decision as to whether or not they trust your applet to not harm his/her system.

TRANSPORT LAYER SECURITY PROTOCOL

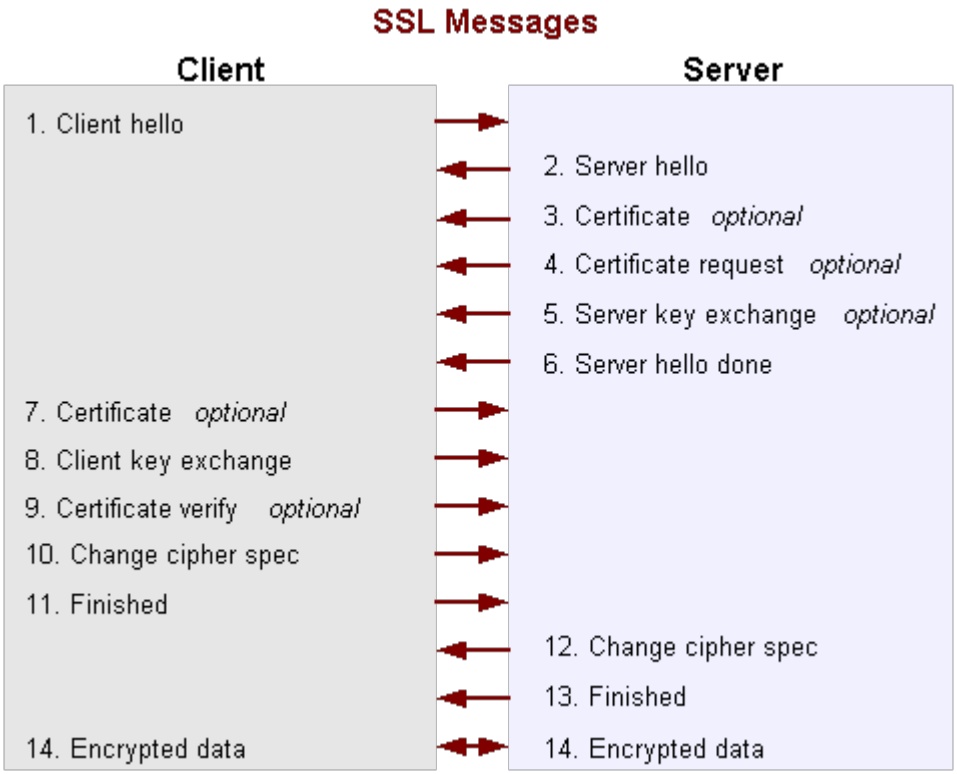
Secure Sockets Layer (SSL) or Transport Layer Security(TLS) is the most widely used protocol for implementing cryptography on the Web. TLS uses a combination of cryptographic processes to provide secure communication over a network. TLS provides a secure enhancement to the standard TCP/IP sockets protocol used for Internet communications. The secure sockets layer is added between the transport layer and the application layer in the standard TCP/IP protocol stack. TLS provides authentication, privacy, and data integrity.

Communication using TLS begins with an exchange of information between the client and the server. This exchange of information is called the TLS handshake.

The three main purposes of the TLS handshake are:

- Negotiate the cipher suite
- Authenticate identity (optional)
- Establish information security by agreeing on encryption mechanisms

There are a sequence of 10 messages which are exchanged during TLS handshake.



1. **Client hello** - The client sends the server information including the highest version of TLS it supports and a list of the cipher suites it supports. The cipher suite information includes cryptographic algorithms and key sizes.

2. **Server hello** - The server chooses the highest version of TLS and the best cipher suite that both the client and server support and sends this information to the client.

3. **Certificate** - The server sends the client a certificate or a certificate chain. A certificate chain typically begins with the server's public key certificate and ends with the certificate authority's root certificate. This is used whenever server authentication is required.

4. **Certificate request** - If the server needs to authenticate the client, it sends the client a certificate request. In Internet applications, this message is rarely sent.

5. **Server key exchange** - The server sends the client a server key exchange message

6. **Server hello done** - The server tells the client that it is finished with its initial negotiation messages.

7. **Certificate** - If the server requests a certificate from the client, the client sends its certificate chain

8. **Client key exchange** - The client generates information used to create a key to use for symmetric encryption. For RSA, the client then encrypts this key information with the server's public key and sends it to the server.

9. **Certificate verify** - When this message is used, the client sends information that it digitally signs using a cryptographic hash function. When the server decrypts this information with the client's public key, the server is able to authenticate the client.

10. **Change cipher spec** - The client sends a message telling the server to change to encrypted mode.

11. **Finished** - The client tells the server that it is ready for secure data communication to begin.

After this a secure channel is established between the client and the server and they can securely transfer data.

The Java Secure Socket Extension (JSSE) enables secure Internet communications. It provides a framework and an implementation for a Java version of the SSL and TLS protocols and includes functionality for data encryption, server authentication, message integrity, and optional client authentication. Using JSSE, developers can provide for the secure passage of data between a client and a server running any application protocol, such as Hypertext Transfer Protocol (HTTP), Telnet, or FTP, over TCP/IP.

The JSSE standard API is available in the javax.net, javax.net.ssl and javax.security.cert packages and covers:

- Secure (SSL) sockets and server sockets.
- Factories for creating sockets, server sockets, SSL sockets, and SSL server sockets. Using socket factories you can encapsulate socket creation and configuration behavior.
- A class representing a secure socket context that acts as a factory for secure socket factories.
- Key and trust manager interfaces (including X.509-specific key and trust managers), and factories for creating them.
- A class for secure HTTP URL connections.
- A public key certificate API compatible with JDK 1.1-based platforms.

A keystore is a database (usually a file) that can contain trusted certificates and combinations of private keys with their corresponding certficiates. A truststore contains the certificates of the entities whom the peer trusts.

A *certificate* is a digitally signed statement from one entity (person, company, etc.), saying that the public key (and some other information) of some other entity has a particular value.

A certificate contains

- Subject
- Issuer name
- Public key of the subject
- Signature of the issuer
- Validity period

The tool used to create and manage keystore is keytool. keytool is a key and certificate management utility. It enables users to administer their own public/private key pairs and associated certificates for use in self-authentication (where the user authenticates himself/herself to other users/services) or data integrity and authentication services, using digital signatures. It also allows users to cache the public keys (in the form of certificates) of their communicating peers.

HAMMING CODE

A Hamming code is a linear error-correcting code. Hamming codes can detect up to two simultaneous bit errors, and correct single-bit errors; thus, reliable communication is possible when the Hamming distance between the transmitted and received bit patterns is less than or equal to one. By contrast, the simple parity code cannot correct errors, and can only detect an odd number of errors.

In mathematical terms, Hamming codes are a class of binary linear codes. For each integer $m > 2$ there is a code with parameters: $[2m - 1, 2m - m - 1, 3]$. The parity-check matrix of a Hamming code is constructed by listing all columns of length m that are pair wise independent.

General algorithm

The following general algorithm generates a single-error correcting (SEC) code for any number of bits.

1. Number the bits starting from 1: bit 1, 2, 3, 4, 5, etc.

2. Write the bit numbers in binary. 1, 10, 11, 100, 101, etc.

3. All bit positions that are powers of two (have only one 1 bit in the binary form of their position) are parity bits.

4. All other bit positions, with two or more 1 bits in the binary form of their position, are data bits.

5. Each data bit is included in a unique set of 2 or more parity bits, as determined by the binary form of its bit position.

    1. Parity bit 1 covers all bit positions which have the least significant bit set: bit 1 (the parity bit itself), 3, 5, 7, 9, etc.

    2. Parity bit 2 covers all bit positions which have the second least significant bit set: bit 2 (the parity bit itself), 3, 6, 7, 10, 11, etc.

    3. Parity bit 4 covers all bit positions which have the third least significant bit set: bits 4–7, 12–15, 20–23, etc.

4. Parity bit 8 covers all bit positions which have the fourth least significant bit set: bits 8–15, 24–31, 40–47, etc.

5. In general each parity bit covers all bits where the binary AND of the parity position and the bit position is non-zero.

The form of the parity is irrelevant. Even parity is purest mathematically, but either even or odd can be used.

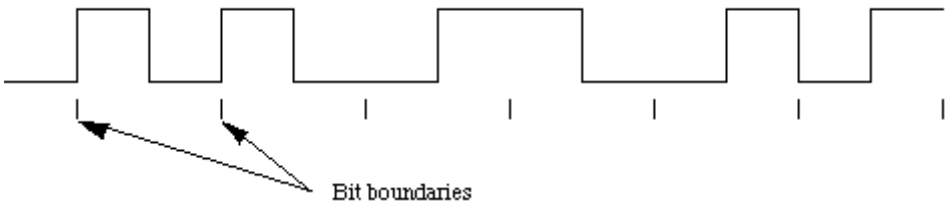
MANCHESTER ENCODING

Manchester encoding (first published in 1949) is a synchronous clock encoding technique used by the physical layer to encode the clock and data of a synchronous bit stream. In this technique, the actual binary data to be transmitted over the cable are not sent as a sequence of logic 1's and 0's. Instead, the bits are translated into a slightly different format that has a number of advantages over using straight binary encoding.

In the Manchester encoding shown, logic 0 is indicated by a 0 to 1 transition at the centre of the bit and logic 1 is indicated by a 1 to 0 transition at the centre of the bit. Note that signal transitions do not always occur at the ‘bit boundaries’ (the division between one bit and another), but that there is always a transition at the centre of each bit. The Manchester encoding rules are summarized below:

Original Data	Value Sent
Logic 0	0 to 1 (upward transition at bit centre)
Logic 1	1 to 0 (downward transition at bit centre)

The following diagram shows a typical Manchester encoded signal with the corresponding binary representation of the data (1,1,0,1,0,0) being sent.



The waveform for a Manchester encoded bit stream carrying the sequence of bits 110100.

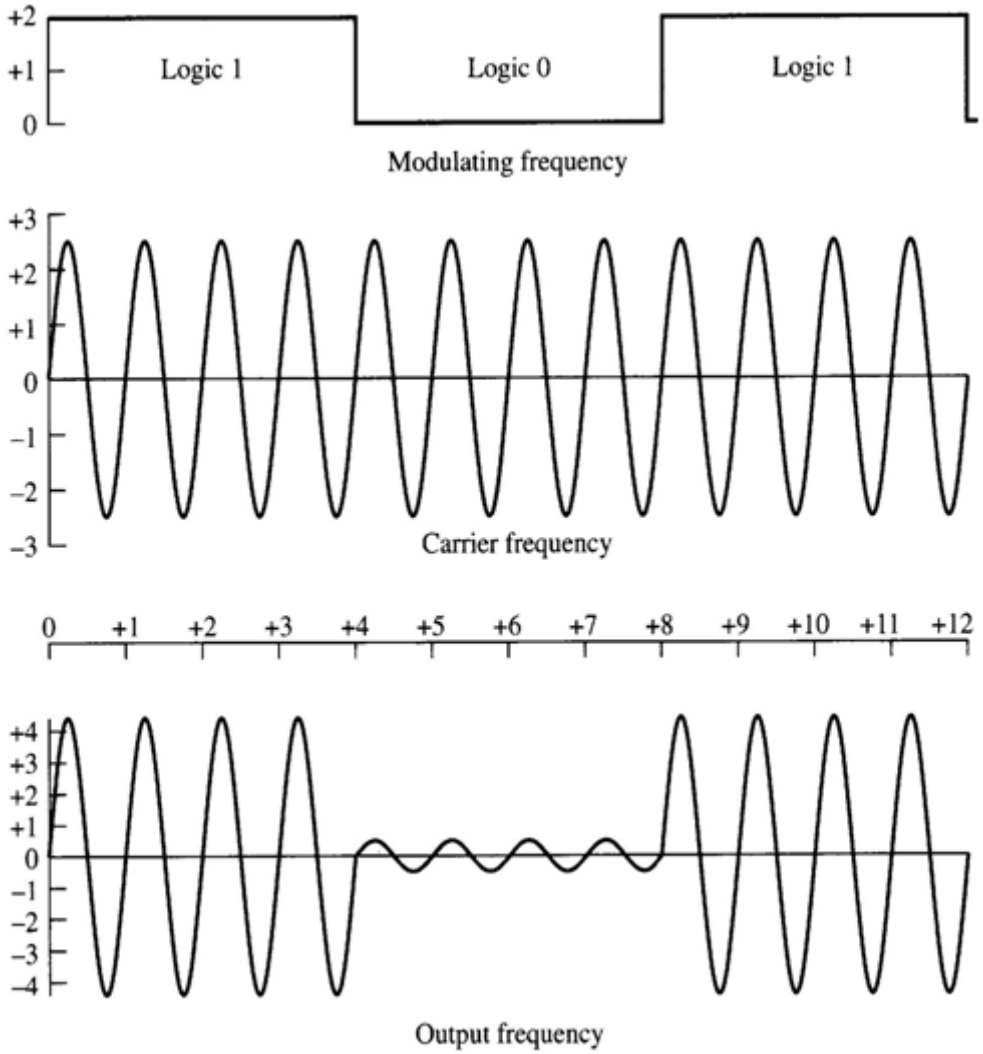
Note that signal transitions do not always occur at the 'bit boundaries' (the division between one bit and another), but that there is **always** a transition at the centre of each bit.The encoding may be alternatively viewed as a phase encoding where each bit is encoded by a postive 90 degree phase transition, or a negative 90 degree phase transition. The Manchester code is therefore sometimes known as a **Biphase Code**.

Example of Manchester Encoding

The pattern of bits " 0 1 1 1 1 0 0 1 " encodes to " 01 10 10 10 10 01 01 10".

AMPLITUDE SHIFT KEY

In this method the amplitude of the carrier assumes one of the two amplitudes dependent on the logic states of the input bit stream. A typical output waveform of an ASK modulator is shown in the figure below. The frequency components are the USB and LSB with a residual carrier frequency. The low amplitude carrier is allowed to be transmitted to ensure that at the receiver the logic 1 and logic 0 conditions can be recognised uniquely.



DISCRETE FOURIER TRANSFORM

The Fourier Transform transforms a function in to its frequency domain representation.

The sequence of N complex numbers $x_0, ..., x_{N-1}$ is transformed into the sequence of N complex numbers $X_0, ..., X_{N-1}$ by the DFT according to the formula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

Frequency domain analysis of audio signals help us to remove noise by passing it through appropriate filters.

BUTTERWORTH FILTERS

The Butterworth filter is one type of electronic filter design. It is designed to have a frequency response which is as flat as mathematically possible in the passband. Another name for it is maximally flat magnitude filter.

The filter function is implemented as

$y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)- a(2)*y(n-1) - ... - a(na+1)*y(n-na)$

where n-1 is the filter order, and a and b are filter coefficients. For our project the coefficients of the filter was selected as follows.

$$a=\begin{bmatrix}1 & 0 & 0\\ 1 & 0 & 0\\ 1 & 0 & 0\\ 1 & 0 & 0\\ 1 & 0 & 0\end{bmatrix} \qquad b=\begin{bmatrix}1 & 2.0061 & 1\\ 0.25 & 0.50 & 0.25\\ 0.25 & 0.50 & 0.25\\ 0.25 & 0.50 & 0.25\\ 0.25 & 0.50 & 0.25\end{bmatrix}$$

ASTERISK

Asterisk is an open source PBX (private branch exchange) that provides all the functionality of high-end business telephone systems, and voicemail services with directory, call conferencing, interactive voice response, call queuing. It has support for most of the VoIP protocols. It is the flexible and extensible telephone system, providing many features that are not yet available in even the most advanced proprietary systems.

It is flexible because it is able to run on different operating systems - Windows, Linux, Mac OS X, Open BSD, etc. It is open source. It is interoperate with traditional telephone technologies. Maintenance is easy.

Asterisk realtime architecture is used to ensures dynamic registration of devices with the Asterisk PBX.

The Asterisk Realtime Architecture (ARA) is a method of storing the configuration files and their configuration options in a database table. The dynamic realtime method is used for things such as user and peer objects (SIP, IAX2), and voicemail which loads and updates the information as it is required. Changes to static information requires a reload just as if you had changed the text file on the system, but dynamic information is polled by Asterisk as needed and requires no reload. MySQL database is configured with the Asterisk server for real time architecture.

SIP COMMUNICATOR

SIP Communicator is a free and open source audio/video Internet phone and instant messenger that supports several popular instant messaging and telephony protocols, including SIP.

In our implementation sip communicator is configured with the client and on successful authentication of the client a sip account is created that is registered with the Asterisk PBX. It is done by editing the sip-communicator.xml configuration file which stores

the sip accounts for that particular sip communicator. Sip communicator reads execution parameters from sip-communicator.xml file. Parameters in this file are currently set so that Sip communicator will use the SIP Services provided by the Asterisk PBX. The registration with Asterisk requires an extension number and a shared secret. The shared secret is that is stored in the sip-communicator.xml file is Base 64 encoded. The encoded secret and the extension number is configured on the sip-communicator.xml file in the device using the signed applets.

6.3 TESTING

Once code has been generated, program testing begins. The testing process focuses on logic internals of the software, ensuring that all statement have been tested and on the functional externals that is conducting test to uncover errors and ensure that defined input will produce actual result that agree with required results.

TESTING OBJECTIVES

Testing is a process of executing a program with the intention of finding an error. A good test case is one that has high probability of finding a yet undiscovered error. A successful test is one that uncovers a yet undiscovered error. Large systems are built of sub systems, which are built out of modules that are composed of procedures and functions. The testing processes should therefore proceed in stages where testing is carried out incrementally in conjunction with system implementation.

UNIT TESTING

This is the first level of testing. Here the different modules of software were tested against the specification produce during the design of the module. Code produced during the coding phase of the software development process and the internal logic of the modules was tested here. Each module was tested individually. The modules of TLS connection, audio generation and recording using Java Sound API, asterisk server configuration, sip communicator configuration were tested and their results noted.

INTEGRATION TESTING

This is a systematic testing for constructing the structure while conducting tests to uncover errors in the interfacing. Here the different modules of the software are combined into the sub system, which are again tested. Verification of the interfaces among system part is tested. The various unit-tested modules of the software were integrated and rigorous integration testing was conducted to make the application remote of any interface errors that may occur during transaction between applications. After the integration of each module, the system was tested rigorously to obtain successful output.

VALIDATION TESTING

Validation testing provides the final assurance that the software meets all functional, behavioral and performance requirements. During the course of validating the software system, to some extent failure occurred and software is changed for the better performance. When the application was made remote of all logical and interface errors, validation testing

was done by inputting dummy data to ensure that the software developed satisfied all the requirements of the user.

SYSTEM TESTING

For every software project there is an inherent conflict of interest that occurs as testing begins. The people who have developed the software are now asked to test the software. Unfortunately these same developers have a vested interest in demonstrating that the program is error remote, that it works according to the customer requirements, that it will be completed in schedule and within budget. From a psychological point of view software analysis and design are constructive tasks. The software engineer creates a computer program, its documentation and data structures. When testing commences, there is a subtle, yet definite attempt, to break the thing the software engineer developed. From the point of view of the developer the testing can be psychologically destructive. The software has undergone the following tests.

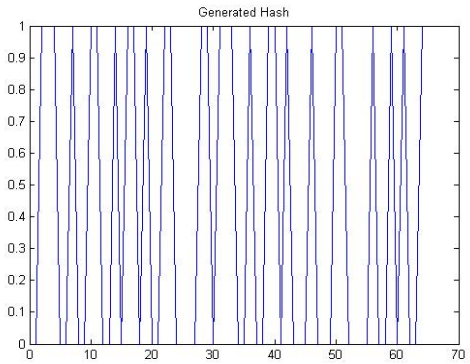
TEST CASE DESIGN METHOD

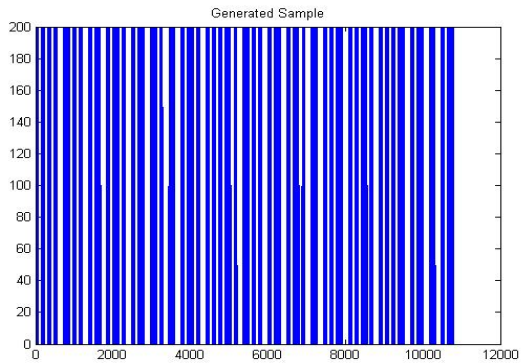
The primary objectives of the test case design method are to derive a set of test that has the highest likelihood of uncovering defects in the software. To accomplish this objective two categories of test case design technique are used: black box testing and white box testing.

7. EXPERIMENTAL RESULTS

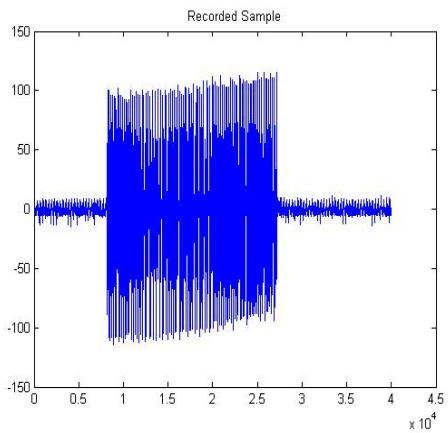
Process of Whisper Generation And Whisper Verification

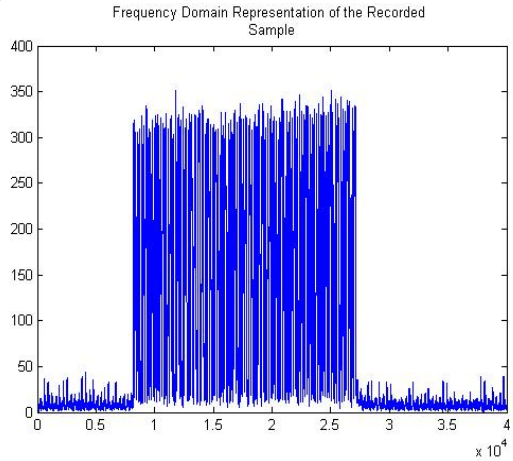
Firstly the hash of the certificate is generated and a carrier wave is modulated to get the audio samples.The audio samples are played through the speaker of the device.



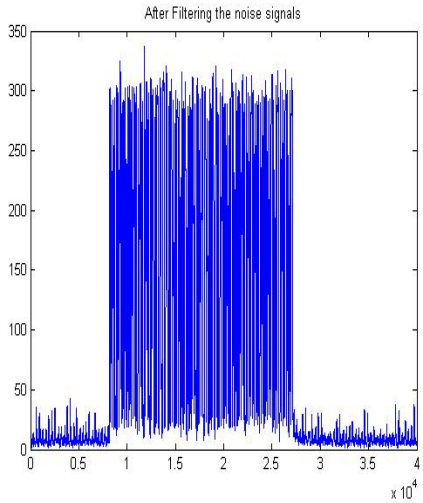


In the TBG the played audio signal is recorded and is processed. Firstly the signal is transformed to frequency domain by finding its fourier transform.

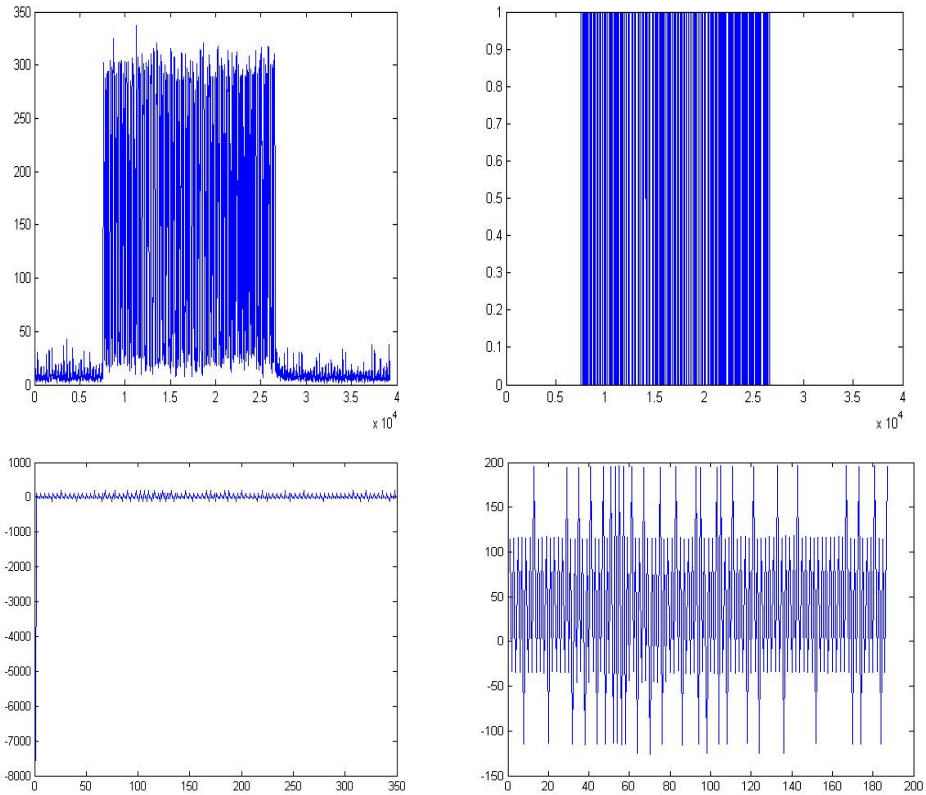




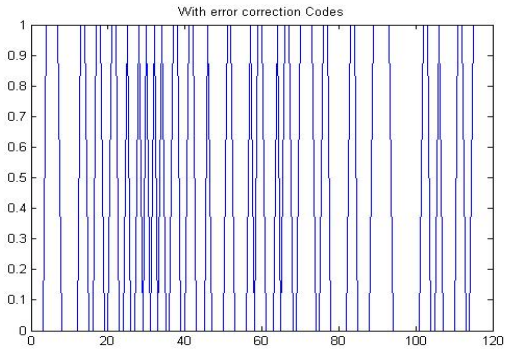
Then unwanted noise signals are discarded by passing it through the filter.

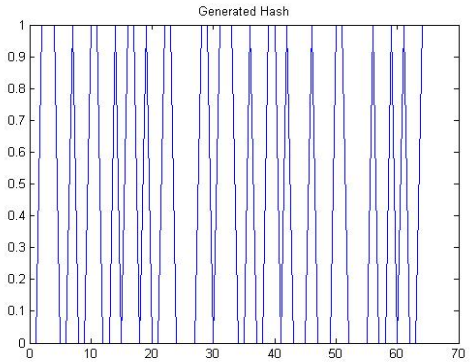


After that the samples are processed to get the hash. The graphs at each step of the decoding process is given below.



After getting the bits encoded with Hamming code, the error correction codes are removed and the orginal hash bits are obtained





Then the two hashes are compared.

8. PERFORMANCE ANALYSIS

Performance analysis is the investigation of a program's behavior using information gathered as the program execute. To optimize the performance of the system that we implemented hamming code so that one bit errors can be detected. We also analysed the performance of the implementation by varying the noise level in the environment. It was found successful to a suceptable level.

9. CONCLUSIONS

The system which enables a wireless guest device to securely register with an enterprise VoIP network is built.

10. USER MANUAL

1. Open the browser in the VoIP-enabled phone.
2. Give the URL written near the gateway.
3. The signed applet gets loaded into the browser.
4. Place the device near the microphone of the TBG.
5. An audio signal is played through the speaker of the device.
6. After authenticating the device, the device gets the extension number.
7. Now the device can avail the VoIP-services of the enterprise till the validity period.

11. REFERENCES

- SMARTPERMIT: A Lightweight and Scalable Architecture for traceable authorizations of guest-devices in Secure 802.11Enterprise Wireless Networks By Jayaraj Poroor and Amit Dhar
- N. Goffee, et al. "Greenpass: Decentralized, PKI-based Authorization for Wireless LANs." 3rd Annual PKI Research and Development Workshop Proceedings. Internet2/NIST/NIH. NISTIR 7122. 26-41, 2004
- Miroslac Milinovic, et al. “Deliverable DS5.1.1: eduroam Service Definition and Implementation Plan.” GÉANT2. Jan 2008.
- Balfanz, D., Durfee, G., Grinter, R. E., Smetters, D. K., and Stewart, P. “Network-in-a-box: how to set up a secure wireless network in under a minute.”, In Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13 (San Diego, CA, August 09 - 13, 2004). USENIX Security Symposium. USENIX Association, Berkeley, CA, 15-15
- www.asterisk.org
- http://sip-communicator.org
- http://dev.mysql.com/doc/
- www.ubuntu.com
- *http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html*
- *http://java.sun.com/j2se/1.4.2/docs/guide/security/jsse/JSSERefGuide.html*