Security System for Mobile Messaging Applications

KTH Information and Communication Technology

Pejman Dashtinejad

Master of Science Thesis

Stockholm, Sweden

TRITA–ICT–EX-2015:2

Security System for Mobile Messaging Applications

Pejman Dashtinejad

8 January 2015

Master of Science Thesis

Examiner
Professor Sead Muftic

Department of ICT
KTH University
SE-100 44 Stockholm, Sweden

TRITA–ICT–EX-2015:2

Abstract

Instant messaging (IM) applications are one of the most popular applications for smartphones. The IMs have the capability of sending messages or initiating voice calls via Internet which makes it almost cost free for the users to communicate with each other. Unfortunately, like any other type of applications, majority of these applications are vulnerable to malicious attacks and have privacy issues.

The motivation for this thesis is the need to identifying security services of an IM application and to design a secure system for any mobile messaging application. This research proposes an E2EE (End-to-End Encryption) approach which provides a secure IM application design which protects its users with better integrity, confidentiality and privacy.

To achieve this goal a research is conducted to investigate current security features of popular messaging applications in the mobile market. A list of requirements for good security is generated and based on those requirements an architecture is designed. A demo is also implemented and evaluated.

Keywords: Mobile, Application, messaging, Chat, Encryption, Security

Acknowledgments

First and foremost, thank you God to give me another chance to complete my higher education and to learn enormous skills and knowledge through all these years of study at the university.

I would like to express my sincere gratitude to Mr. Sead Muftic for his patience and guidelines through the course of this project. His guidelines helped me to comprehend how to formulate my ideas and solutions into a structured design and an acceptable paper work.

I am also very grateful to all the people whom helped me to finalize this work. For all of their sincere cooperation, brainstorming and contribution. God bless you all.

Stockholm, December 2014
Pejman Dashtinejad

Table of contents

List of Figures

List of Tables

List of acronyms and abbreviations

AES	Advanced Encryption Standard
APDU	Application Protocol Data Unit
API	Application Programming Interface
CA	Certificate Authority
CRL	Certificate Revocation List
CSR	Certificate Signing Request
CSRF	Cross-Site Request Forgery
DDoS	Distributed Denial of Service
DES	Data Encryption Standard
DOM	Document Object Model
DoS	Denial of Service
DSA	Digital Signature Algorithm
FIPS	Federal Information Processing Standard
FTP	File Transfer Protocol
GPL	General Public License
GSM	Global System for Mobile communications
GUID	Globally Unique IDentifier
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
IO (I/O)	Input Output
IP	Internet Protocol
IPS	Intrusion Prevention System
IPsec	Internet Protocol Security
IRC	Internet Relay Chat
ISO	International Standard Organization
ISP	Internet Service Provider
IT	Information Technology
LAN	Local Area Network
LCA	Local Certificate Authority
MD5	Message-Digest algorithm 5
MNO	Mobile Network Operator

NFC	Near Field Communication
NFS	Network File System
NIST	National Institute of Standards and Technology
OS	Operating System
OSF	OpenSSL Software Foundation
PCI	Peripheral Component Interconnect
PIN	Personal Identification Number
PIV	Personal Identity Verification
PKI	Public Key Infrastructure
RSA	Rivest, Shamir, Adleman (public key cryptography)
SA	Strong Authentication
SD	Secure Digital
SDK	Software Development Kit
SIM	Subscriber Identity Module
SMS	Short Message Service
SOAP	Simple Object Access Protocol
SSH	Secure Shell
SSL	Secure Socket Layer
SSTP	Secure Socket Tunneling Protocol
TCP	Transport Control Protocol
TLS	Transport Layer Security
TSM	Trusted Service Manager
UPnP	Universal Plug and Playn
URL	Uniform Resource Locator
USB	Universal Serial Bus
UUID	Universally Unique IDentifier
VoIP	Voice over IP
VPN	Virtual Private Network
XML	Extensible Markup Language
XSS	Cross-site scripting

1 Introduction

Mobile chat applications are very popular among Internet users and smartphones' owners. Hundred millions of smartphone owners use chat applications on monthly basis [1]. These chat applications offer the communication free of charge and majority of them are free to install which makes it very appealing for the potential customers.

These chat applications offer different services and built-in features to their users while in majority of the cases, they neglect security aspects of their usages and messages. According to a report provided by Electronic Frontier Foundation* (EFF) majority of these chat applications do not provide enough security for their users [2].

This chapter describes a brief background about the topic, problem definition and goals of this thesis. It also discusses contributions of the author and the scope of the research and its limitations. This chapter ends with the outline of the report and a short description of each chapter.

1.1 Background

Smartphones have become indispensable tools in today's workforce. There are hundred millions of smartphones in the market with different operating systems and capabilities. These smartphones have the capability of installing applications on them in order to do many tasks, such as sending text messages.

Wi-Fi networks have grown dramatically for the last decade and they are almost ubiquitous in daily life. According to a report by quarter 3 of year 2014, there are 2.5 billion mobile broadband subscriptions worldwide and it is predicted that it will be 8.4 billion mobile broadband subscriptions by the year 2020 [3]. This growth and accessibility to an Internet connection has brought opportunities for many Instant Messaging applications to enter a new area of communication and to penetrate traditional telephony communication.

Instant messaging applications are one of the most popular category of applications among the users. The IMs have the capability of sending messages or initiating voice calls via Internet, what makes it almost free for the users to communicate with each other and to share different type of files.

1.2 Problem Definition

Malicious users are always interested to hack servers and reveal information about users in a certain system including celebrities and this happens almost every day in the

* https://www.eff.org/about

Internet world. Unfortunately Mobile instant messaging applications are not an exception [4]. There are many mobile chat applications available for users. Many of these applications claim that they are providing confidentiality, integrity and availability of user's information. However, daily hacking news prove that many developers do not consider security as the primary goal of their applications.

On the other hand, governments are keen on tracking their citizens and forcing more service providers to reveal profiles of their users in the hands of their agents. Furthermore, chat applications providers misuse information of their users. For example, while many chat applications are free to use, they equip the application with built-in processes which track every single movement of their users and they use these type of information to sell to 3rd party advertisement agencies.

1.3 Purpose of the Thesis

The main purpose of this thesis is to design a secure chat application which protects user's confidentiality and privacy. A research was conducted to investigate current security features of several messaging applications on smartphones application stores. The selected instant message applications has been investigated and a list of requirements for the design of a secure chat application was created. Based on different features and requirements, a design has been proposed and a demo is also implemented.

1.4 Goals

In this project different mobile chat applications have been investigated. Based on the threat model of mobile chat applications, different requirements for a secure chat application have been enlisted and a design was also proposed.

The goal of this project is to propose solutions for different security challenges of the current chat applications in the market and to design and implement a secure chat application. The main contributions of this research can be structured into the following four sub-goals:

1. Literature study and related security vulnerabilities in current mobile chat apps
2. To propose a secure mobile chat application architecture
3. A detailed design of secure mobile chat application
4. To implement and evaluate a demo as proof of concept (POC)

1.5 Research Methodology

This thesis is based on design science research methodology which makes it convenient to develop an artifact. In such a methodology [5] a problem is defined and the goals of the research is defined. A research is conducted to investigate current popular mobile

chat applications, their features and threats that they might enforce to their users based on security flaws or bugs. As the next step, the requirements for secure chat application are defined and architecture of a secure chat application is proposed. The final result of the research is a secure mobile chat application architecture. Based on the proposed design, a proof of concept is implemented and analyzed.

1.6 Limitations

In this thesis, the proposal and the design of the secure instant messaging application is based on the client/server model. However with the rise of mesh networks and built in features in smartphones' operating systems such as iOS* which has the built-in ability of "multi peer connectivity", very few peer-to-peer chat applications are available in the market. Peer-to-peer (P2P) chat clients are omitted from this thesis since their architecture and infrastructure is different. Another thesis can be derived to focus only on such peer-to-peer mobile chat applications and investigation of their security vulnerability and design flaws.

1.7 Structure of the thesis

Chapter 1 gives the introductory information such as the background of the topic, problem statement, what the researcher is going to solve, which methodology he is using to achieve the objectives of the thesis, what are the scope and limitations of the thesis and what has not performed during the course of this thesis.

Chapter 2 presents relevant background and contains a brief study of mobile chat applications and security concerns of such applications for the user. In this chapter, different security aspects of a generic mobile application are explained and some of the existing and popular chat applications are explored from the security point of view.

Chapter 3 different requirements and features that a secure mobile chat application system must have are discussed and solutions to each problem are presented. Based on the solutions, a secure mobile chat application system is designed and an architecture is proposed in the last section of this chapter.

Chapter 4 presents practical implementation and challenges of such an architecture in the real scenario. As the proof of concept a small demo is presented at the end of this chapter as well

Chapter 5 explains evaluation and analysis of the secure chat application.

Chapter 6 explains the results, objectives that have been achieved, and future work of the research which concludes the research.

* iPhone Operating System iOS is mobile operating system designed by Apple and runs on its products such as iPhone and iPad which is currently active on more than a billion devices. Apple also runs its own appstore which gives the user the freedom of options between hundreds of thousands applications based on iOS operating system

In this report the terms IM (Instant Messaging) and chat application (chat app) will be used interchangeably. It is also good to mention that through this thesis, the only focus is on mobile chat applications installed on a mobile operating system. Therefore no other types of clients, such as personal computers, are addressed.

2 Background

Penetration of the smartphones at the global scale is very high. A mobility report from Ericsson* company states that by the year 2020 smartphone subscription will be 6.1 billion and 90% of world population over 6 years old will have access to a mobile phone [3]. There are many reasons behind this fast growth, but the most important are more powerful hardware and decrease in the cost of manufacturing. These factors make smartphones more affordable in the developing markets, such as China and India, the countries with more than one billion population.

Users of smartphones have access to millions of applications in their application stores although there are different smartphone operating systems for smartphones, two are dominant, Android† and iOS.

Each of these operating systems have a big application ecosystem which gives the option to billions of users to choose the desired service and use them right from a device in their pocket. As of July 2014, there is 1,300,000 apps available for Android in Google Play Store‡ and 1,200,000 of apps available for iOS Apple§ store [6].

2.1 Smartphone Ecosystem

The ecosystem of application stores provides a chance for everybody including freelance developers as well as companies to publish their applications to the world via these stores. After the application gets approved by the store, it can be discoverable from anybody who has access to the Internet to surf and search through the app store.

Every industry can benefit from such ecosystem. Many industries including insurance, banking, healthcare and even government agencies are getting benefits of smartphones' penetration and Internet access to better serve their customers and stakeholders.

Among the popular mobile applications, IM applications have hundred millions of users worldwide [7] and there are many companies that provide such a service to their users. All these chat applications compete with each other to get higher penetration rate among the users and to defeat other companies in the market. These IM chat providers try to create better user interfaces and offer more features for their respective users.

2.2 Security Services for Mobile Instant Messaging

In order to evaluate any chat application from the security point of view, relevant threats to such application should be identified and described. In the following sections a brief description about different security aspects are explained.

---

* http://www.ericsson.com/
† https://www.android.com/
‡ https://play.google.com/store
§ https://itunes.apple.com/us/genre/ios/id36?mt=8

Security has three key aspects: confidentiality, integrity and availability [8]. Confidentiality ensures that certain type of information can be accesses by authorized parties. Integrity means information can be modified only by intended and authorized parties. Availability means that information is accessible to authorized parties at appropriate times [9].

2.2.1 Confidentiality

Confidentiality means messages which are exchanged by two parties through a communication channel should be readable only to the intended parties. In order to achieve such a goal, encryption is the mechanism that provides confidentiality between two parties. A message is encrypted by a cryptographic technique and this encrypted message can only be readable by the intended party.

2.2.2 Cryptography

Cryptography is the practice and study of techniques which are used to secure a communication between two entities while the third party (adversary) exist. Cryptography helps to create an environment or medium channel in which confidentiality, integrity, authentication of the user and non-repudiation are supported.

There are two major types of algorithms in cryptography. One is symmetric key cryptography and the second is public key cryptography. In symmetric key encryption both parties use a shared key to encrypt and decrypt the messages [10]. In this method, because the adversary does not have the shared key, even if he intercepts the communication channel, he just receives encrypted messages and is not able to decrypt them.

Public key cryptography, is also known as asymmetric cryptography, uses an algorithm which needs two separate keys: one key is private key and the other key acts as the public key of the user, However there is a link between these two keys which is generated by an algorithm for a user [11].

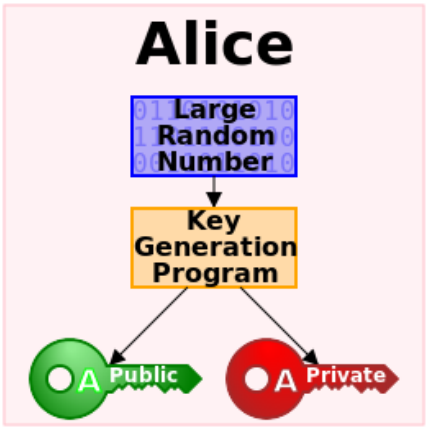

Figure 1 Generating key pairs based on public key (asymmetric) algorithm*

When public key cryptography is used, the application generates two keys. Public key can be shared or broadcast to the whole world, but the knowledge of the private key remains secret. In this method, one party uses the second party's public key and encrypts the message and sends it over to the second party. The second party receives the encrypted message and opens it with its own private key. If any malicious hacker intercepts the communication channel, he just receives encrypted messages and is not capable to decrypt them, because he does not have the private key of the recipient.

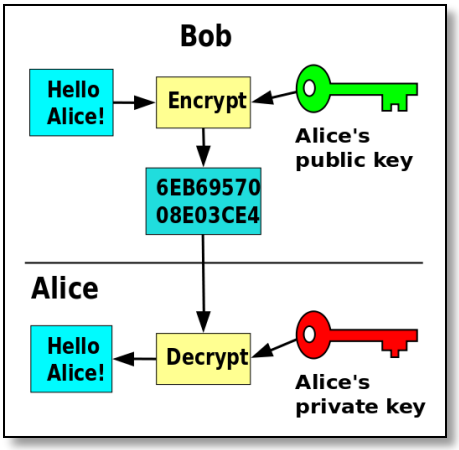

Figure 2 Encryption and decryption in public key (asymmetric) algorithm†

* The picture is public domain and can be used without any condition
† The picture is public domain and can be used without any condition

2.2.3 Authentication

Authentication is one of the most important aspects of security, where an entity should identify itself before or during the communication. This avoids any type of attack or malicious activity by which a malicious user impersonates the user and identifies himself as the real user to the server.

There are two types of authentication schemes known as weak authentication and strong authentication. Weak authentication (one factor authentication) means that the entity uses only one type of identity credential such as a PIN* or password-based authentication. It is considered a weak mechanism because it is prone to many attacks including brute force attacks. A brute force attack is type of attack that the malicious user tries as much as passwords to finally finds out the one which matches the chosen password of the user.

Strong authentication is usage of typically a challenge-response cryptography. In this scheme the client needs to prove his identity and verify himself to the server with multiple factors. There are different practices to perform such authentication such as one-time passwords (OTP) and certificate-based authentication (CBA). In one-time password, a shared secret key is stored on a device that the entity has, and the system issues one-time passwords based on this shared secret key.

CBA is using asymmetric cryptography which provides public-private key cryptography. In this method, each user has a unique digital signature and this digital signature can be used to verify the true identity of the user. Digital signatures are explained later on in this chapter.

2.2.4 Integrity

Integrity insures that a message has not been edited or changed during the transfer of it between entities. An attacker can eavesdrop the communication channel and modify the message or even replace the message with a new one. Hashing is a mechanism to achieve such a goal in the world of information security. A cryptographic hash function is a function which maps an encrypted message to a fixed size length integer. A hash function is one-way function, meaning that if somebody has an output of a hash, it cannot be reversed

* Personal Identification Number

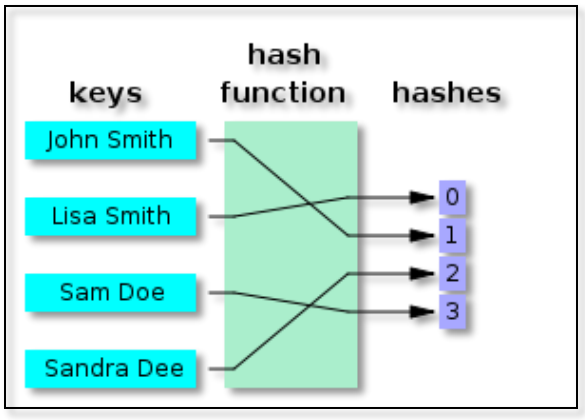

Figure 3 Hash Function*

As an example, one entity encrypts a message and generates a hash out of the encrypted message. The recipient uses the same hash function to generate a hash out of the received encrypted message. The recipient compares the two hashes. If both of the hashes are the same, it means the message is original. There are different algorithms and hash functions to achieve such process.

* The picture is public domain and can be used without any condition

2.2.5 Privacy

In a digital world privacy of a user means the power to select what type of information to be shared or be accessible by other entities including governments, service providers, and even other applications. Privacy of a user is related to and based on a metadata that an application can collect and send to a second or third party.

Unfortunately privacy is one of the factors that is getting sacrificed in the current mobile application environment. Many applications are free of charge, but the service providers grab metadata of the user and send these information to their servers even sometimes without the knowledge of the users. Metadata means any type of data which includes different information about a user, such as location, name, contact information, creator of data, contact list, type of operating system, etc….

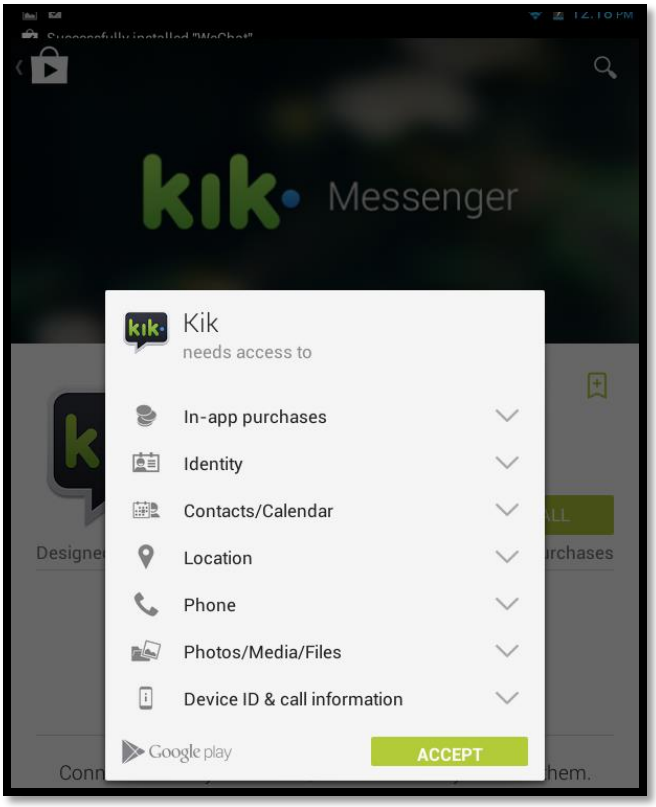

Figure 4 IM access request to users Metadata and information

2.2.6 Non-Repudiation

Non-repudiation means that a mechanism in which the entity who has sent the message cannot deny that. A malicious user can sometimes generate a message and send it on behalf of an entity to another party without knowledge of the entity. Non-repudiation solves this problem by introducing a mechanism which proves that the source of a message is a person that claims it. Like the real world where people can have signatures or even seals to seal or sign a contract, in the digital world and in public-private key ecosystem, there are digital signatures which can be related to different entities and individuals can sign their messages with their dedicated digital signature.

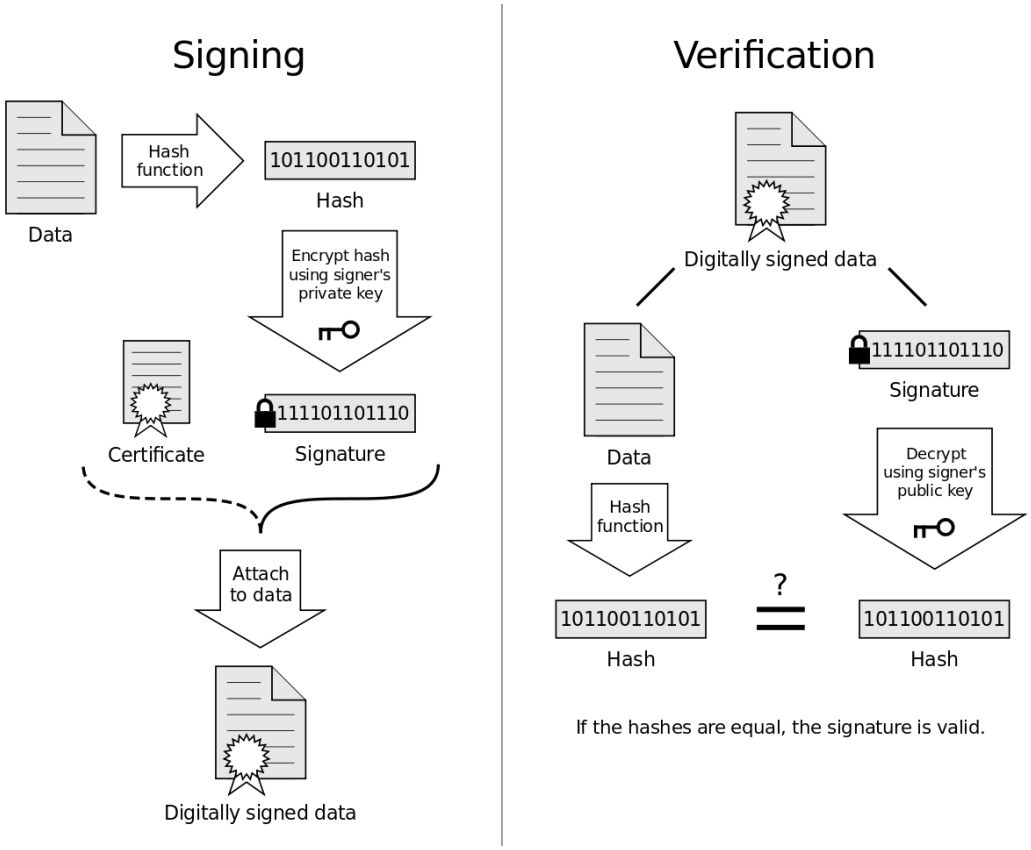

Figure 5 Signing and verification by Digital Signature*

* The picture is public domain and can be used without any condition

2.3 Mobile Chat Applications

As it was mentioned in previous sections, there are many chat applications in the mobile market. Recently some chat applications have started to distinguish themselves in the market by calling themselves as "secure chat application". Normally in these types of chat applications, they are self-claimed by their providers that they have prioritized security and privacy of their users at the first place. According to number of downloaded applications from the application stores, these so-called security applications have few millions of users each in the market and they are not among the top popular chat applications at all. Although majority of chat applications use various types of encryption, unfortunately in most of the cases, the IM owners' servers issue the keys or have access to message keys to decrypt them!

Unfortunately some chat applications have proprietary protocols or architecture and they are not public or open source. These apps cannot be examined by the developer's community or security experts what makes it harder for evaluation. Therefore the only way is to trust the providers' wordings regarding their claims.

Another way is to perform extensive reverse engineering* or penetration testing† routines in order to evaluate security levels of these applications. However they are still black box testing meaning that the security researchers do not have access to the source code to see how the applications works with 100% accuracy.

In order to investigate IM apps' security features, several IM applications have been selected based on their popularity. In the following sections, a range of mobile chat applications will be briefly introduced. Four different chat applications have been selected due to factor of popularity, business orientation, and self-claimed security providing services.

2.3.1 WeChat

Wechat‡ is the third popular messaging application in the market available in different platforms including iOS and Android. It supports sending voice, video, pictures and text messages. Unfortunately, the architecture of the application is proprietary and it has its own protocol, but it has an official page for developers [12] regarding WeChat SDK§. WeChat does not provide end-to-end encryption meaning that encryption methods that is used is based on public key encryption, but the user needs to trust the WeChat servers.

* Reverse Engineering in any process to understand the design or knowledge of a computer program
† Any activity to attack a computer system with the intention of finding vulnerabilities both as design flaw or software bug.
‡ http://wechat.com/
§ Software Development Kit is a set of software development tools which helps a developer to create their programs

2.3.2 Voxer

While majority of mobile chat applications market themselves for consumers, Voxer* is more business oriented and market itself as a mobile messenger for teams. Voxer also has desktop version which can be installed on a normal PC. Voxer has free and paid version.

Voxer is a proprietary application and there is not good documentation about the structure of their servers. Based on their official website, they claim that they use military graded encryption for the encryption of the messages and also they use secure communication channel using TLS† to send messages between parties.

2.3.3 Wickr

Wickr‡ has some unique features which makes it very appealing for users [13]. For example one of the features that they offer is self-destruct messages meaning that as soon a user reads a message, the message will be wiped from the recipient's mobile phone. Based on their official website they claim that the application removes all the metadata and they do not upload users' contact book to their servers. While they claim that they don't have a backdoor§, the software is not open source and there is no way for an external auditor to verify this claim! Wickr uses end-to-end encryption (E2EE). In E2EE, the encryption and decryption of messages happens at the users' mobile phone.

2.3.4 Viber

Viber** is one of the most popular free chat applications with hundred millions of users all around the world. A research was conducted [14] by the UNH Cyber Forensics Research and Education Group in order to investigate security of Viber application. This research revealed explicitly that Viber is not secure at all in many cases. The research has revealed that media files such as pictures or videos which are transferred between the users have no encryption and the data is stored on the Viber server unencrypted which can be accessed without any authentication mechanism! These vulnerabilities gives the ability to a malicious user to simply launch MitM†† attacks and capture unencrypted data either over the wired or wireless networks. Viber does not support end-to-end encryption

---

* http://voxer.com/

† Transport Layer Security is a protocol to provide privacy assurance between two communicating parties. TLS is the successor to previous protocol which was called Secure Socket layer protocol (SSL)

‡ https://wickr.com/

§ Is a method to bypass normal authentication of the system.

** http://viber.com/

†† Man-in-the-Middle Attack is kind of attack in which a malicious user intercepts the communication channel and listens to the conversation of two parties. For example a malicious hacker can run such an attack and sniff and gather the network packets between two computer nodes in a network.

2.4 Related Work

A research [15] conducted in 2011 has evaluated the security of nine mobile chat applications from authentication point of view. Based on the experiments that they have done, the result of their research shows that there are major security flaws in most of their tested applications and makes the applications prone to different types of attacks. Unfortunately in this research they are not considering other aspects of security and also they have not proposed any design to formulate a secure chat application.

A research was conducted [16] by two researchers to investigate security of iMessage* reveals that in spite of the fact that the owner of the application claims that the application has high encryption standards and it is an end-to-end encryption but there is still a possibility for the server owners to read their customer messages in case they want to perform such a task!

A research [17] was conducted to investigate security of ChatON† instant messaging application from forensics‡ point of view which means what a malicious user can retrieve out of a mobile phone if he or she has physical access to the phone. In this research they revealed that by physical access to a mobile phone and proper software, the database of messages for both sent and received messages and their timestamps could be retrieved.

A series of researches [18] reveals that unfortunately many social media applications, including chat applications, breach privacy of the users and are prone to different type of vulnerabilities for their users. Vulnerabilities include revealing plain-text passwords or storage of private information on the servers which can be revealed to non-authenticated users.

The main shortcoming of all mentioned articles is that neither of them describe the structure of a chat application and also do not have any suggestion for a design of proper security in an IM application. Majority of the reports, articles or researches regarding security of chat applications are addressing solely on evaluation of their security, based on experimentations and lab setups. Even in research cases where researchers have dived more into the structure of a chat application, they have not proposed any solution or a specific design for a secure chat application system.

* iMessage is the Proprietary messaging application of Apple company which is built in into its mobile operating system and MAC OS X which is Apple operating system for personal computers.
† ChatOn is proprietary instant messaging of Samsung company
‡ Forensics is the science of finding legal evidence in computer systems or digital storage device

2.5 Summary

In this chapter an introduction to the mobile IM applications was presented. Different security issues and background information have been explained and different chat applications have been researched in order to investigate their current security features that they offer to their users.

The table below shows the status of security in each application based on different criteria. As it has been mentioned, majority of chat applications are closed source and proprietary with no good documentation, what restricts external auditors and security experts to investigate the reliability of their respected applications.

Name	E2EE*	Open Source	Secure channel	Secure storage†
WeChat	NO	NO	YES	NO
Voxer	NO	NO	YES	NO
Wickr	YES	NO	YES	YES
Viber	NO	NO	YES	NO

Table 1 Messaging application score board

Please note that the data presented in the table might be changed from the time of research due to the fast release pace and the new updates of the chat applications. It should be also considered that there are cases [19] in which a chat application uses a certain type of security on one operating system while it does not support it on another platform. For example, a chat application might encrypt information of the user profile in Android platform while, it is not performing the same process in iOS. The reason behind this is because the customer pool in one platform is much bigger, so the provider tends to push the updates faster for that specific popular operating system rather than for the other ones!

* End to End Encryption

† It means if the application stores the information of the user such as messages encrypted on the device. This is important because in case the mobile phone gets stolen or any malicious user who has physical access to the device can retrieve those unencrypted information.

3 Secure Mobile IM Design

In this chapter the design of a secure mobile chat application will be explained. This architecture has to meet the requirements of confidentiality and integrity of messages, as well as privacy of users.

3.1 Mobile Chat Requirements

Before designing any system, a list of requirements and features should be created. The followings are the requirements of such a system which provides good enough security for a private chat session between the two users*.

- Providing Session-Level Security (SLS), as a unique key is generated for each session. All the messages exchanged in previous and also future sessions cannot be read by intruder.
- Each message has its own separate key which brings better security for each single message.
- All local data on the mobile phone should be encrypted by a separate key derived from the PIN entered by the user. This PIN never leaves mobile application
- Support for offline messaging. In case one of the users is not online, the other user can still send him messages. These messages should be encrypted and stored online, so whenever another user comes online, they can be transferred to the recipient
- All the sessions or information exchanged back and forth between any entities should be encrypted to avoid malicious attacks.
- The design provides integrity of the messages which assures the recipient that the message has not been modified on the way.
- The design should provide non-repudiation to assure the message is from the real sender

* This list can be extended and it should be considered as a basic list which fulfills confidentiality, integrity and privacy of the users.

3.2 Architecture Proposal

In a secure system which uses a cryptography scheme to provide security services for the users, one of the most important questions is who has the key to decrypt the encrypted message. As a requirement to reduce the role of the server and make it zero knowledge, the best solution is end-to-end encryption, where encryption and decryption are performed at the end nodes. For example, chat application on the mobile phone encrypts the message, sends it over to the server, and server sends it to the recipient. Now the message can only be decrypted in the local device of the recipient and the server cannot read messages, because the server does not have the required keys to decrypt the message!

The proposed secure chat application has the end-to-end encryption (E2EE) method of communication. E2EE means that the end-points are responsible to encrypt data and all encrypted data transferred between different users of the system is only readable to the users who have the key. In this architecture even the server does not have the keys, so if the server gets compromised or forced to reveal the keys of the messages, it will be irrelevant, because the server has zero knowledge of the keys and the encrypted messages. Of course limitations of CPU power and battery drain might be one consideration to be cautious about choosing type of cryptography algorithms.

The network architecture of the proposed design can be based on multi-tiered structure in which servers can be physically separated. However, for the sake of this thesis, all the servers are stored on one physical server. The server is support two services. The CA* server is responsible for certificate issuance. The second server which is IM server is responsible for messaging service which includes offline storage of encapsulated messages in case the recipient is not online. Below is the list of security services which the proposed architecture provides to the user:

- Registration of users
- Certificates management
- Authentication of the users
- Secure sessions with the IM server
- Keys exchange between users
- Encryption of messages
- Decryption of messages
- Offline secure messages' storage. This happens when the second party is not online. These offline messages are encrypted and not readable by the server

* Certificate Authority

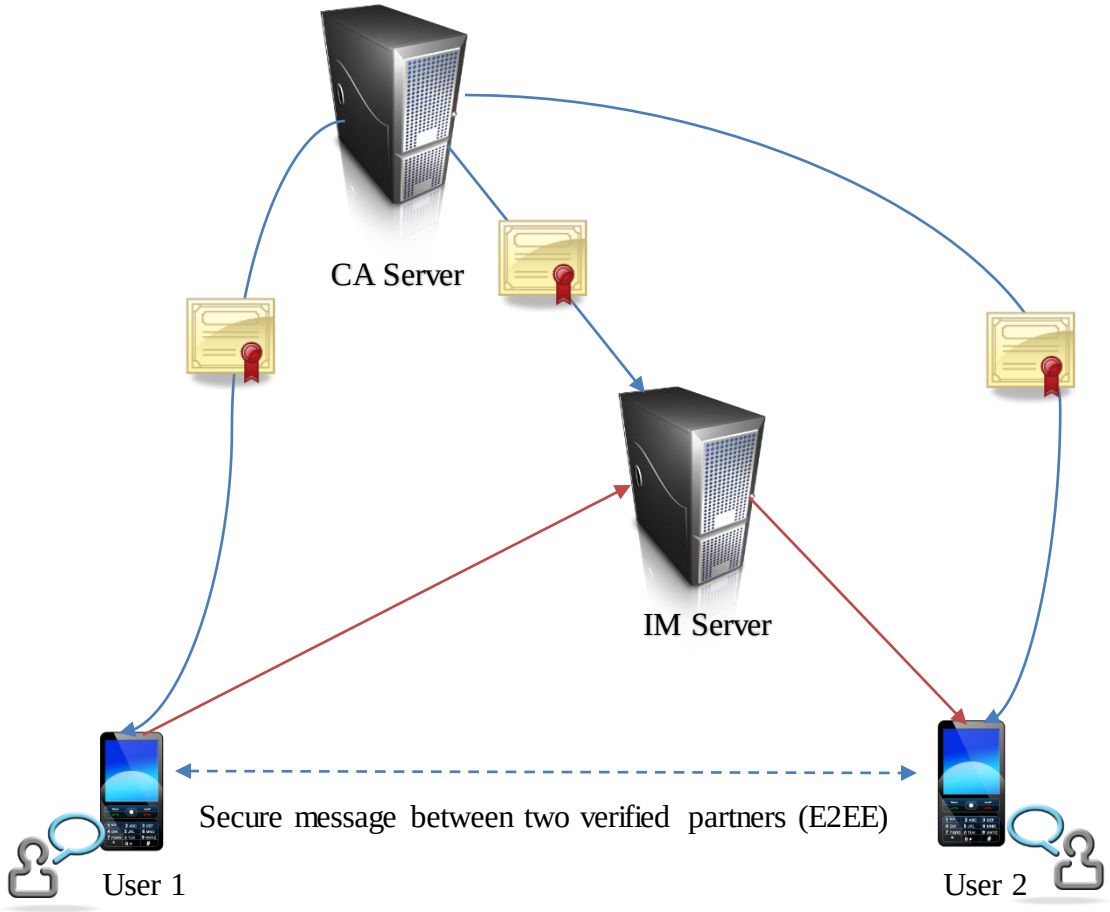

Figure 6 Overview of the architecture

3.3 Secure IM Application Stages

Based on the requirements of the design in the previous section, our secure chat application has different distinguished stages. Each stage has different steps which is explained in details in this chapter.

Stage 0 - Application setup: In this stage the user downloads the application to the mobile phone from an online application store and simply installs the application in the phone.

Stage 1 - Signup process: At this stage each new user should register himself to the IM server and afterward the user should go through the process of certification exchange. This process gives the ability to both parties (the server and the mobile client) to secure their communication channel via SSL/TLS in the future connections. At this stage the

application on the user's mobile phone generates a local profile to store certificates as well.

Stage 2 – Partner verification: This stage is necessary before any two users start chatting with each other and it includes two steps. The first step is to establish a secure connection to the IM server and the second step is to exchange the keys between the two parties planning to exchange encrypted messages later on.

Stage 3 – Message exchange: In this stage three steps are performed: message creation, message encryption, and message decryption. Each step has different processes which should be performed by the chat application.

3.4 Stage 0. Application Setup

In this stage, the user downloads the application from the application store and installs it in the mobile phone. Different mobile operating systems have different application stores. Any application which is needed to submit to the application store goes to the process of review, so a user should consider those applications available on the application store as reviewed and verified apps.

In a mobile environment, when the user downloads an app, the app is installed automatically and there is no extra action of the user needed to install it.

3.5 Stage 1. Sign-up Process

The main goal of this step is to register a new user and make him ready to exchange secure messages. Sign-up process has three different steps which are the following:

- Local profile creation
- Certificate issuance
- Registration with the IM server

3.5.1 Local Profile Creation

User starts the application. A window will show up which asks the user to enter his email address and a 6 digit PIN number. The email which user provides in this step is stored in the local profile. The PIN can be stored and retrieved by the native system supported by the hosting mobile operating system which is secure enough. The system uses this PIN to encrypt local profile that provides another layer of security. In this scenario if the malicious user has access to the actual user mobile phone, he cannot retrieve those information, because the whole profile is encrypted and the hacker does not know about the PIN.

3.5.2 Certificates Issuance

At this step, user generates a key-pair and stores them in the local profile. This key pair is protected, since the complete local profile is encrypted by the PIN. Now the user sends certification request to the CA server. In this request user sends its own generated public key and CA server returns user's certificate together with its own certificate. User's certificate and CA's certificate are added to the local profile.

3.5.3 Registration with IM Server

In this step, user enters PIN and activates the application. The application sends the email of the user and the user's certificate which is stored locally in the user profile to the IM server. IM server sends the activation link to the user's email. User should open up his email and click on the link which has been sent by the IM server to activate his account. As the last part, the IM server sends its own certificate to the user. Now the user is ready to create a secure communication channel via TLS to the server.

3.6 Stage 2. Partner Verification

In this stage two steps happen. First, the chat application automatically creates a secure communication channel to the IM server via normal TLS handshake process. As part of requirements which was session-level security, each session should have a separate unique key.

The second step is called "Users key exchange". In this step, user1 sends the email address of user2 to the IM server. IM server will return the certificate of the user2 to the user1. Clearly the user2 should have been registered to the IM server in the previous stage.

3.6.1 Establishing Secure Connection to IM Server

User launches the application and enters the PIN. The application automatically starts to establish a secure connection to the IM server via a TLS handshake process [20]. During the handshake both the client and the server decide on the type of certificate, cryptography ciphers and the protocol version [21]. Only at the time of agreement, the handshake process is over, and the client can establish a secure communication channel with the server.

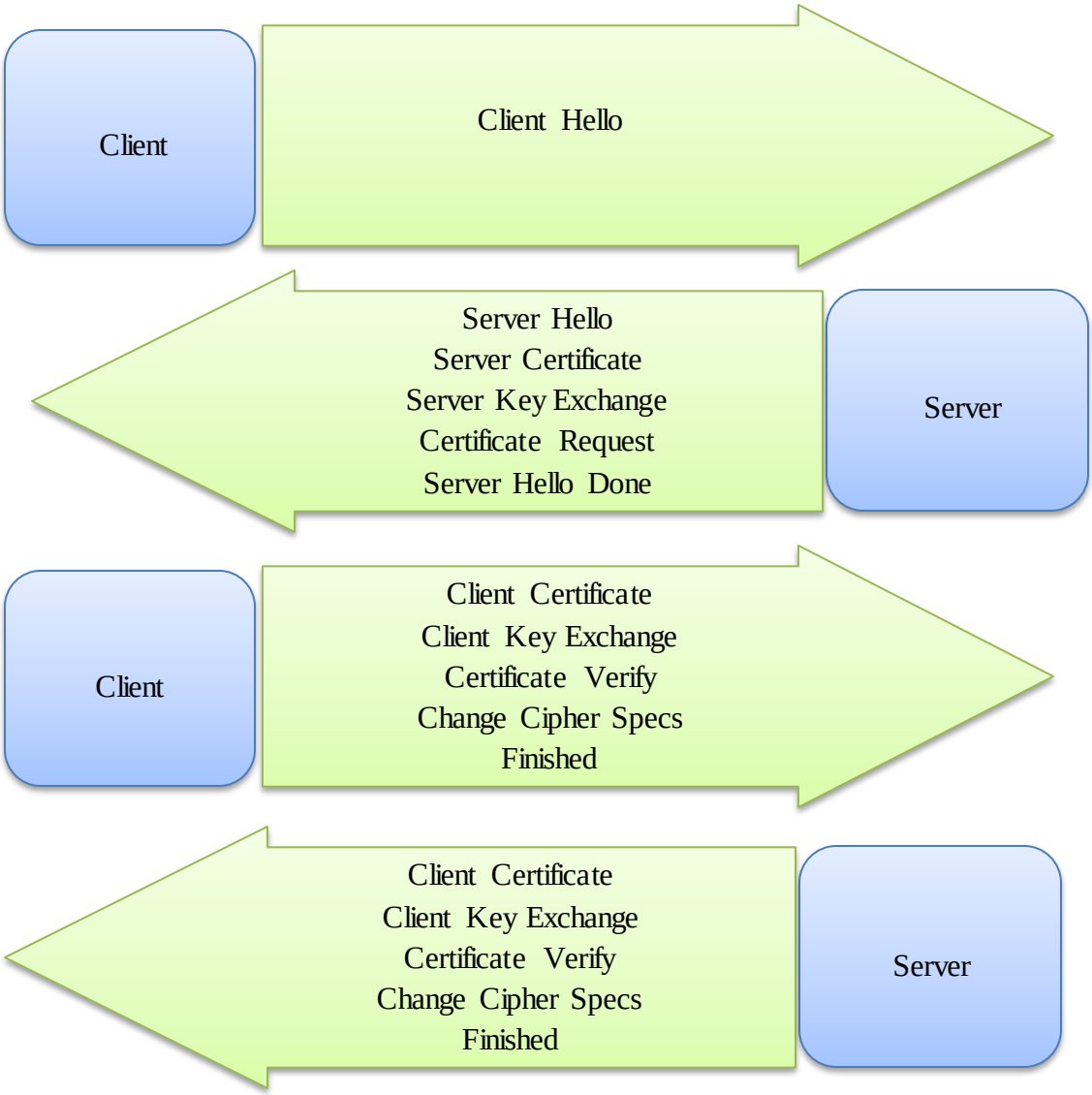

Figure 7 Typical Flow of TLS Protocol

3.6.2 Users' Key Exchange

In this step, user1 sends the email of user2 to the IM server. The assumptions here are firstly that user1 knows the email address of user2, because the email address of each person is used as the unique identity of each user and secondly user2 should have already registered himself with the server.

Now, IM server replies to the user1 by sending the certificate of user2. The application on user1 mobile phone will add user2 certificate to the local profile for the future use. Because the user1 has the certificate of user2 and it is stored in the local profile securely for the upcoming message exchanges, this step is not needed to be performed. This step happens only if the user1 does not have the certificate of the recipient.

3.7 Message Exchange

This stage has three distinguished steps. The first step is "message creation" includes all the processes needed to encrypt the message and to encapsulate all the necessary information needed for the recipient. The second stage is "message sending" which involves all the processes needed to send an encapsulated message to the desired receiver. The last stage is "message receiving" which includes all the processes needed to receive and decrypt the message securely.

Figure 8 Overview of active conversations on the server

3.7.1 Message Creation

It this step, the sender types the message. The application generates a random message encryption key and encrypts the message by the generated key. Next, a message hash is calculated. The chat application encrypts this hash with the private key of the sender. As the final step the application encrypts the message key with user2 (receiver) public key. The encapsulated message is ready to go to the next step. The encapsulated message consists of encrypted message, encrypted message key, and secured hash.

Figure 9 Encapsulated Message Structure

3.7.2 Message Sending

In this step, sender sends the encapsulated message to the IM server. If the receiver is online (meaning that user2 has currently a secure connection to the IM server), IM server forwards immediately the encapsulated message to the receiver. In the case the recipient is offline, the encapsulated message is stored at the IM server. As soon as the receiver becomes online, it will be forwarded.

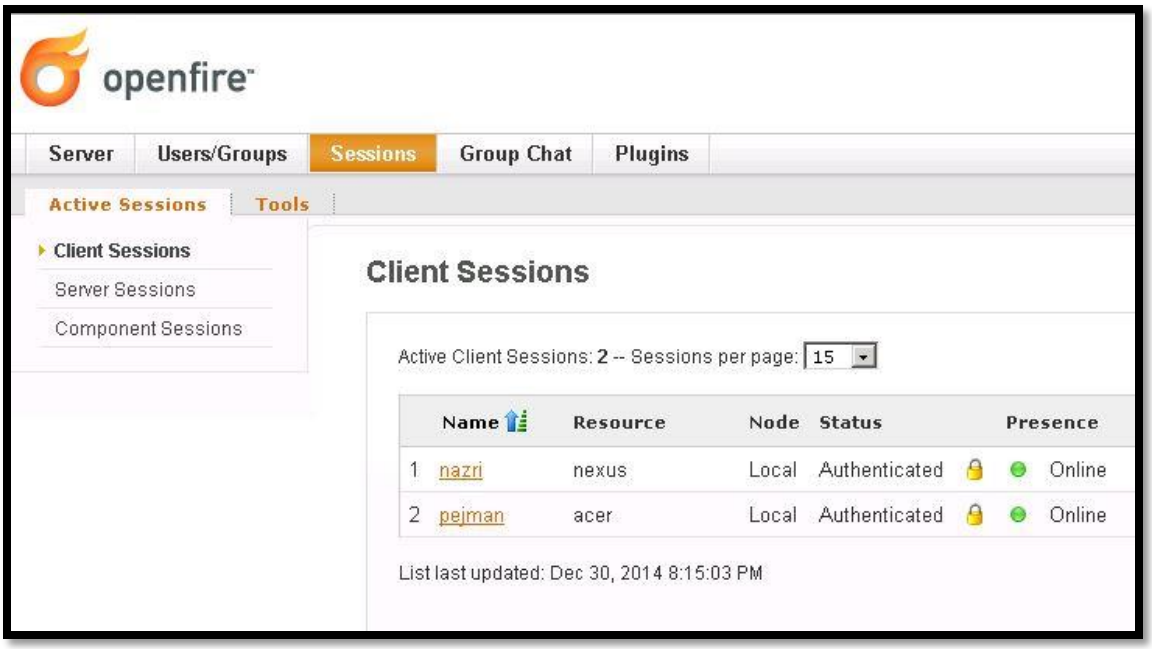

Figure 10 Screen shot of IM server revealing current online users

3.7.3 Message Receiving

In this step, the receiver gets the encapsulated message. Next, application decrypts the encrypted hash using sender's public key. Next the application calculates the hash of the encrypted message and compares it with the received hash to verify the integrity of the message. If both hashes are the same, it means the message has not been exchanged or modified in the communication channel.

As the next step, the application decrypts the message key using the receivers' private key. Now when the application knows the key, it will use this key to decrypt the message and the message is readable to the recipient at the end of this process.

4 Implementation - Demo

Based on different aspects of the proposed design and its features described in the previous chapter, the required hardware and software have been selected and configured in order to prepare the infrastructure for the demo. The deployment of such design requires implementation of an IM server and a mobile application based on proper protocols. In this chapter different aspects of the implementation and related experience is discussed.

4.1 Protocol and Code Library

There are two major protocols which compete with each other to be the dominant protocol for instant messaging communication: XMPP* and SIMPLE†. Both of these protocols are open standards. There are many arguments [22] regarding benefit of each protocol, but according to designers and programmers XMPP is better because it uses XML‡ which is better for implementation and deployment of different applications running on different systems. XMPP has many other benefits [23] as well which has made it as one of the biggest open source communities with different open source applications on different platforms. The core features of XMPP have been defined in RFC 3920 [24].

A definite guide to XMPP book [25] has been read in order to understand more about XMPP and also what to consider at the time of design and coding of a chat application. However, this book does not have any section specifically for an Android application development.

Currently there are many code libraries [26] available based on different programming languages in order to build an XMPP-based IM application. The most used and well-known coding library for Android operating system is asmack [27].

4.2 Cryptographic Specifications

In the proposed architecture AES-256 by NIST in FIPS-197 document [28] is used in order to fulfil the registration of the user to the IM and encapsulating the messages.

RSA [29] algorithm with 2048 bit based on the recommendation by NIST [30] has been selected for creation of key pairs on the mobile client. As for integrity support, the HMAC-SHA1-512 has been selected [31][32].

* Extensible Messaging and Presence Protocol
† SIP for Instant Messaging and Presence Leveraging Extensions
‡ Extensive Markup Language

4.3 Server Implementation

There are many options to choose for an IM server [33]. However, the priority was to select an IM server which has the least license restrictions, broad popularity and ease of installation. As the result, Openfire* 3.9.3 has been installed on the hosting Windows server.

Openfire is open source and has several nice features. It can be installed on both the Windows machine and Linux machine. It supports "offline messaging" in case the recipient is not online. If the receiver is not online, the message is stored at the IM server database until the user becomes online, when the offline messages are sent to the user and removed from the server.

As the choice of database for the messages storage MySQL† has been selected and for the ease of implementation XAMPP‡ has been installed. XAMPP in an easy to install Apache§ distribution containing MySQL, PHP and Perl.

The IM server is installed on an Intel** Core i3, 2.5 GHz with 2 GB RAM machine and the hosting operating system is Microsoft†† Windows Server R2 2008.

4.4 Mobile Implementation

For the sake of testing and presentation, two Android devices were used. One device was the Nexus S smart phone running Android version 4.1.2 and the second device was Acer Iconia B1-720 tablet with Android version 4.4.2 installed. Both devices had the capability of connecting to the Wi-Fi connection in order to connect to the Internet.

The environment for programming was the Eclipse IDE‡‡ installed on a Windows 8.1 based laptop. ADT (Android Development Tools) bundle§§ was installed and configured as the plugin for Eclipse [34]. There is a built-in emulator inside Eclipse to run and test the application on different emulated Android devices, but that Emulator is very slow. For convenience, a direct connection of an Android device via USB cable is always advised for testing and debugging.

* http://www.igniterealtime.org/projects/openfire/index.jsp
† http://www.mysql.com/
‡ https://www.apachefriends.org/download.html
§ http://www.apache.org/
** www.intel.com
†† http://technet.microsoft.com/library/dd349801
‡‡ https://eclipse.org/home/index.php
§§ http://developer.android.com/tools/help/adt.html

4.5 Demo Scenario

In the demo, two registered users to the IM server are intended to send non-encrypted and encrypted messages. First, user1 needs to add user2 as his contact in the chat application.

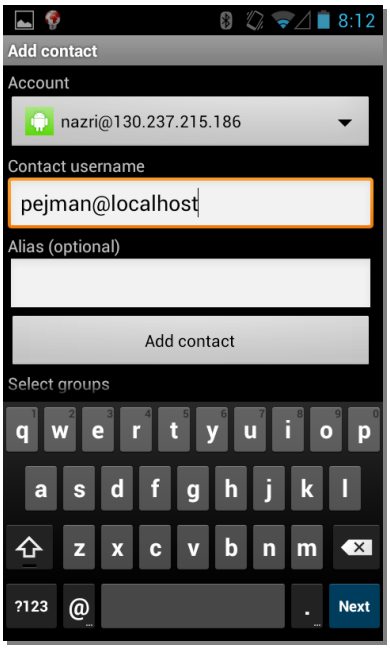

Figure 11 Screen Shot of adding a user as contact

After adding the intended user to the contact list, user1 can see the list of contacts with their current status. In the following screen shot, user2 light is green which means he is online and available.

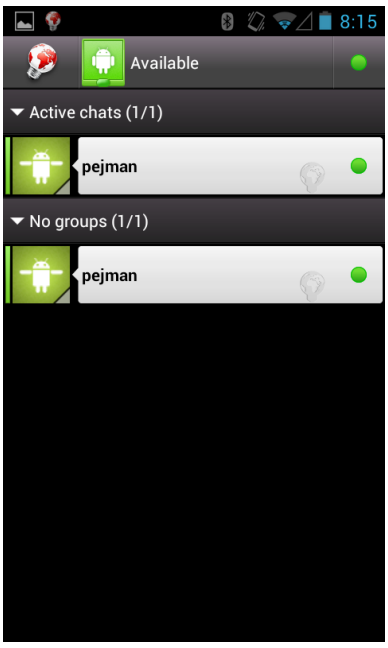

Figure 12 Screen shot of online contacts in the IM app

From the application window shown in the previous figure, user1 selects an online contact and starts a chat session with him.

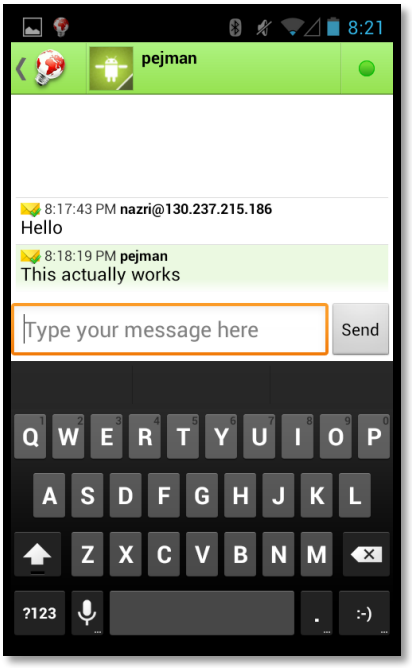

Figure 13 Screen shot of IM application's chat window between two users

So far the conversation is not encrypted, therefore it is fully readable by the administrator from the admin panel of the IM server.

Figure 14 Screen Shot of admin panel for a non-encrypted conversation

Now, as soon as user1 enables the encryption and sends the encrypted messages, it is readable by user2 but it is not readable by the administrator and it is shown as encrypted garbage characters to the viewer of admin panel.

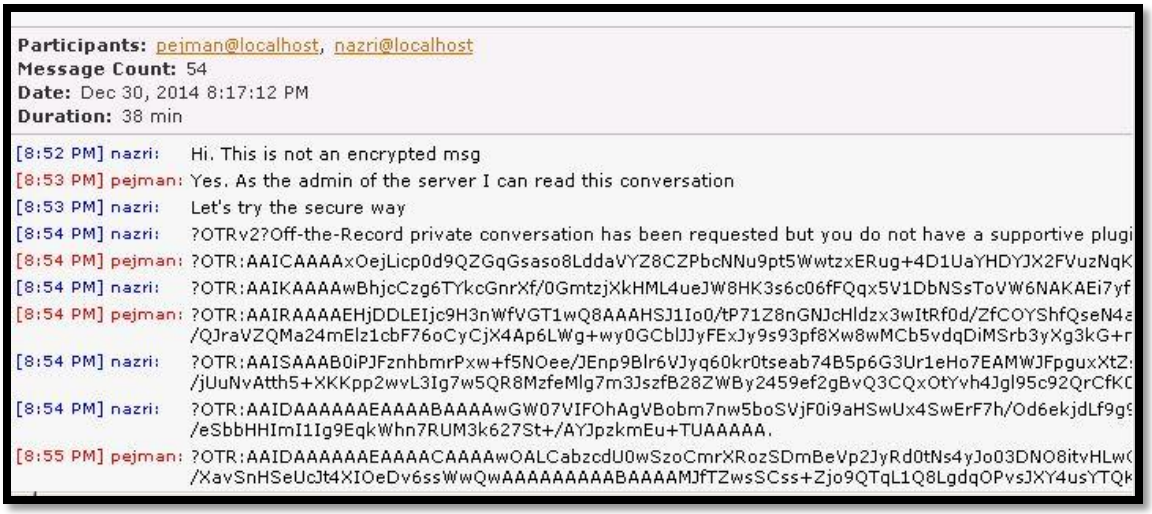

Figure 15 Screen Shot of admin panel for an encrypted conversation

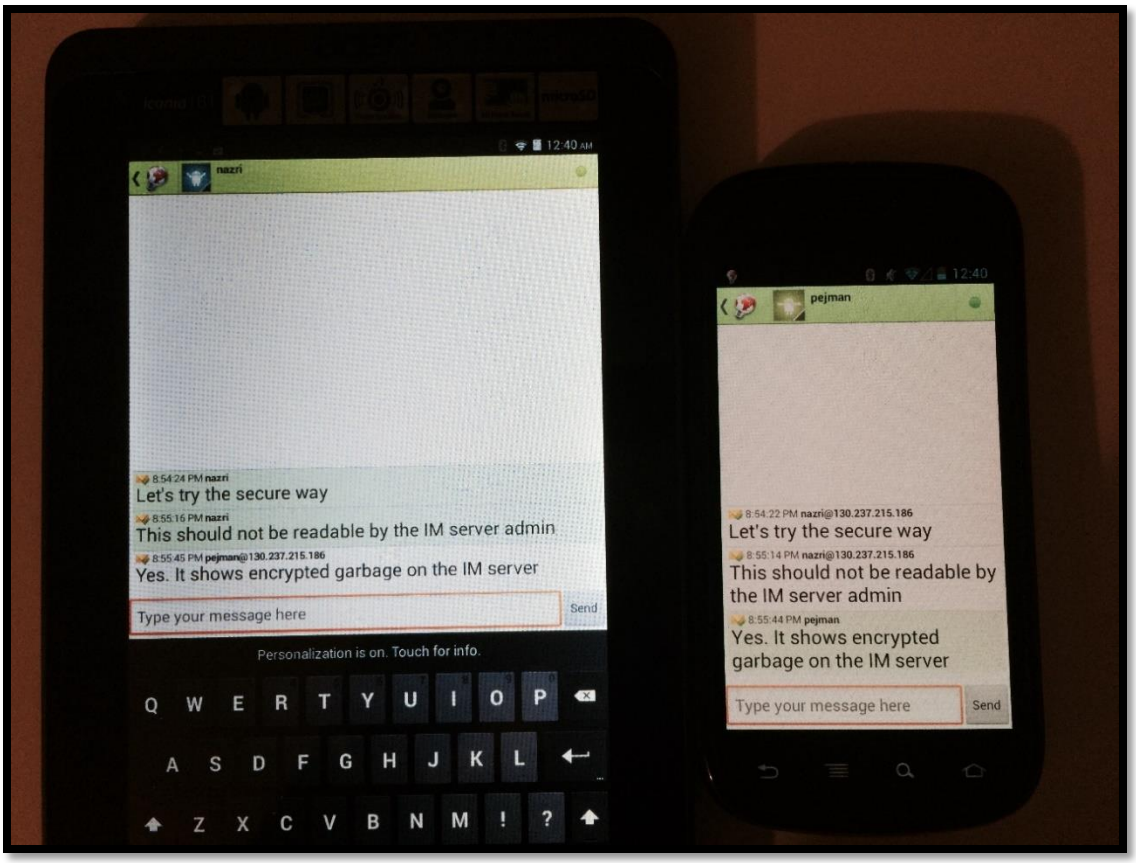

Figure 16 Screen Shot of Message Exchange between two Android devices in encrypted mode

5 Results and Analysis

In this chapter the results and evaluations of the design based on different security aspects are presented. The main goal is to see how many features have been achieved based on security requirements and specifications, enlisted in the beginning in order to provide good security for users.

The objective of the implementation was to show a small demo as a proof of concept that the proposed infrastructure is functioning. However it should be mentioned that, in order to create a successful secure chat application, design of a secure architecture is only the first aspect. The second aspect is coding and programming of the design which needs high programming skills.

5.1 Analysis of the Architecture

In section 3.1 the list of basic requirements for a secure chat application had been created. The following table shows the achievements during the demo.

Status	Requirement	Description
☑	SLS support	Perfect SLS was achieved during the implementation
☑	Message Encryption	Messages are encrypted and not readable by the server.
	Local Profile Encryption	All the information on the profile of the user is encrypted and only the user with the PIN can decrypted the information. Due to lack of time, this feature was not presented for the demo.
	Offline Messages	In case the recipient is not online, messages are stored at the IM server and will be moved to the recipient as soon as, he becomes online. Due to lack of time, this feature was not implemented and presented for the demo
☑	Session Encryption	All sessions between different entities should be encrypted. It is done by TLS connection. During the demo application creates a secure TLS connection to the server.
☑	Message Integrity	The design supports integrity of messages which assures that messages have not been modified or changed during transfer. Integrity has been implemented though via OTR protocol
☑	Non-repudiation	To assure that a message is sent by the real sender. Due to difficulty of establishing and configuring a CA server, this feature was shown with different method which was to verify by user's SHA1 fingerprint

Table 2 Secure chat application requirements

6 Conclusions and Future Work

In this thesis the main contribution was to define requirements and parameters of a secure mobile chat application in which provides to users different aspects of security, such as confidentiality, integrity of messages and privacy. An architecture was proposed with details about implementation of such a design and a demo was also created. However due to time limitations, this research can be extended to provide more features to the chat application and better implementation of the infrastructure.

In this chapter the conclusion and some of the writer's suggestions are mentioned in order to open opportunities for future researchers and students to continue this project.

6.1 Conclusions

The proposed architecture for a secure mobile chat application provides confidentiality, integrity and privacy for users who want to send text messages to each other without the need for extra hardware or physical tokens. Users can be confident that nobody, even not the provider of the service, can read their messages. Even in the case that mobile phone reaches wrong hands, no readable information can be extracted from the physical memory of the phone.

During the implementation of the architecture, several difficulties were encountered. Some of these challenges have been solved after extensive trials and workarounds and some of them remain. Some of the difficulties were regarding configuration of separated software. For example, during the course of this thesis it turned out that configuration of a PKI infrastructure even as the simplest form (tier-one) is not an easy task and needs hands-on experience. Appendix A has some details about these challenges and the proper hints.

6.2 Future Work

In order to design very effective secure system different factors should be considered. Although the goal of this thesis was merely to secure messages between two clients, there are some other factors that might be considered for future research such as usability of the application or scalability of the servers or how the design effects the cost of implementation. In order to complete this research the following topics are suggested for future researchers:

Performance: the proposed design has not been analyzed to investigate how it effects the performance of the mobile phone. How much it will consume CPU power and how it affects battery drain. Different security algorithms or encryption ciphers might need more computational power and support from the underlying operating system.

Privacy and Profiling: Unfortunately in many cases, this totally depends on the mobile operating system version and its platform. Lots of unnecessary information is being sent by free or paid versions of IMs to their servers which in many cases the user has no idea about it . Further research should be conducted to see how it is possible to restrict or remove those metadata which is sent by any chat application to the server.

To avoid **reverse engineering** attacks, it is always a great practice to automatically clearing all cache data with any change in the IM application activity. For example, when the application goes to the background. This means to avoid many mistakes in the implementation and coding of the application. OWASP* has a great standard and checklist document [35] for this purpose.

Implementation of the "**group chat**" feature might be considered as well. In group chat, more than two parties can join and chat with each other. Security of a group chat is more complicated mostly when the chat application has capability of offline messages and needs more considerations in the design phase of the architecture which could be a topic of a new research.

For the sake of authentication, **smartcards** can be used and connected to the mobile phones which can be considered as future work of this research. In the proposed design, it was the goal not to use any hardware token or separate device attached to the mobile phone, because they can be lost that brings less usability to the user.

* Open Web Application Security Project

References

[1] "Most popular global mobile messenger apps 2014 | Statistic," *Statista*. [Online]. Available: http://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/. [Accessed: 02-Dec-2014].

[2] "Secure Messaging Scorecard," *Electronic Frontier Foundation*. [Online]. Available: https://www.eff.org/secure-messaging-scorecard. [Accessed: 02-Dec-2014].

[3] "Ericsson Mobility report," Ericsson, P4, Nov. 2014.

[4] "Hacking attack on accounts connected with popular messenger app - Australia Network News (Australian Broadcasting Corporation)," *Hacking attack on accounts connected with popular messenger app*. [Online]. Available: http://www.abc.net.au/news/2014-06-19/an-asia-cyber-crime/5537194. [Accessed: 23-Dec-2014].

[5] K. Peffers, T. Tuunanen, R. Marcus A., and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, 2007.

[6] "• Number of apps available in leading app stores 2014 | Statistic." [Online]. Available: http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. [Accessed: 03-Dec-2014].

[7] "• Number of mobile messaging users worldwide 2014 | Statistic." [Online]. Available: http://www.statista.com/statistics/369260/mobile-messenger-users/. [Accessed: 03-Dec-2014].

[8] "NIST SP 800-12: Chapter 1 Introduction." [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-12/800-12-html/chapter1.html. [Accessed: 02-Dec-2014].

[9] C. P. Pfleeger and S. L. Pfleeger, "1.3 The Meaning of Computer Security - Security in Computing, 4th Edition." [Online]. Available: http://flylib.com/books/en/4.269.1.16/1/. [Accessed: 02-Dec-2014].

[10] "Introduction to Cryptography: Principles and Applications - Second Edition. Hans Delfs, Helmut Knebl," 2007. Chapter 2, P. 12 [Online]. Available: http://books.google.se/books?id=Nnvhz_VqAS4C&pg=PA11&. [Accessed: 02-Dec-2014].

[11] C. Kaufman, R. Perlman, M. Speciner, and M. Speciner, *Network Security: Private Communication in a Public World*. Prentice Hall PTR, 2002. chapter 2, section 2.5 public key cryptography

[12] "WeChat Developers," *WeChat API Documentation*. [Online]. Available: http://dev.wechat.com/wechatapi/documentation. [Accessed: 03-Dec-2014].

[13] "Wickr | Top Secret Messenger," *Wickr | Top Secret Messenger*. [Online]. Available: https://wickr.com/. [Accessed: 03-Dec-2014].

[14] "unhcfreg | Viber Security Vulnerabilities: Do not use Viber until these issues are resolved." [Online]. Available: http://www.unhcfreg.com/#!Viber-Security-Vulnerabilities-Do-not-use-Viber-until-these-issues-are-resolved/c5rt/BB4208CF-7F0A-4DE1-92A4-529425549683. [Accessed: 28-Nov-2014].

[15] S. Schrittwieser, P. Kieseberg, L. Manuel, P. Fruhwirt, M. Mulazzani, M. Huber, and E. Weippl, "Guess Who's Texting You? Evaluating the Security of Smartphone Messaging Applications." [Online]. Available: https://www.sba-

research.org/wp-content/uploads/publications/ndss2012_final.pdf [Accessed: 02-Oct-2014].
[16] C. Cattiaux, "iMessage Privacy," 17-Oct-2013. [Online]. Available: http://blog.quarkslab.com/imessage-privacy.html. [Accessed: 03-Dec-2014].
[17] A. Iqbal, A. Marrington, and I. Baggili, "Forensic artifacts of the ChatON Instant Messaging application," in *2013 Eighth International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE)*, 2013, pp. 1–6.
[18] "unhcfreg," *UNH Cyber Forensics Group Reveals Smartphone App Issues Affecting 968 Million*. [Online]. Available: http://www.unhcfreg.com/. [Accessed: 03-Dec-2014].
[19] "SQLCipher has 100M+ Mobile Users (Thanks to WeChat!) | The Guardian Project," *SQLCipher has 100M+ Mobile Users (Thanks to WeChat!)*. [Online]. Available: https://guardianproject.info/2013/12/10/sqlcipher-has-300-million-mobile-users-thanks-to-wechat/. [Accessed: 03-Dec-2014].
[20] T. D. <tim@dierks.org>, "The Transport Layer Security (TLS) Protocol Version 1.2." [Online]. Available: http://tools.ietf.org/html/rfc5246. [Accessed: 26-Dec-2014].
[21] "SSL and TLS: A Beginners Guide - ssl-tls-beginners-guide-1029." [Online]. Available: http://www.sans.org/reading-room/whitepapers/protocols/ssl-tls-beginners-guide-1029. [Accessed: 31-Dec-2014].
[22] "XMPP vs SIMPLE: The race for messaging standards," *InfoWorld*, 23-May-2003. [Online]. Available: http://www.infoworld.com/article/2678727/operating-systems/xmpp-vs-simple--the-race-for-messaging-standards.html. [Accessed: 31-Dec-2014].
[23] O. Ozturk, "Introduction to XMPP protocol and developing online collaboration applications using open source software and libraries," in *2010 International Symposium on Collaborative Technologies and Systems (CTS)*, 2010, pp. 21–25.
[24] E. P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core," 2004.
[25] P. Saint-Andre, K. Smith, and R. Tronon, *XMPP: The Definitive Guide Building Real-Time Applications with Jabber Technologies*. O'Reilly Media, Inc., 2009.
[26] "Libraries – The XMPP Standards Foundation." [Online]. Available: http://xmpp.org/xmpp-software/libraries/. [Accessed: 31-Dec-2014].
[27] "Flowdalic/asmack," *GitHub*. [Online]. Available: https://github.com/Flowdalic/asmack. [Accessed: 31-Dec-2014].
[28] Information Technology Laboratory, "Digital Signature Standard (DSS)," National Institute of Standards and Technology, NIST FIPS 186-4, Jul. 2013.
[29] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," *Commun ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.
[30] "NIST Special Publication 800-57, Recommendation for Key Management Part 1: General (Revision 3) - sp800-57_part1_rev3_general.pdf." [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf. [Accessed: 27-Dec-2014].

[31] "FIPS 198-1, The Keyed-Hash Message Authentication Code (HMAC) - FIPS-198-1_final.pdf." [Online]. Available: http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf. [Accessed: 27-Dec-2014].

[32] H. Krawczyk, R. Canetti, and M. Bellare, "HMAC: Keyed-Hashing for Message Authentication." [Online]. Available: https://tools.ietf.org/html/rfc2104. [Accessed: 27-Dec-2014].

[33] "Servers – The XMPP Standards Foundation." [Online]. Available: http://xmpp.org/xmpp-software/servers/. [Accessed: 26-Dec-2014].

[34] "Installing the Eclipse Plugin | Android Developers." [Online]. Available: http://developer.android.com/sdk/installing/installing-adt.html. [Accessed: 31-Dec-2014].

[35] "Category:OWASP Application Security Verification Standard Project - OWASP." [Online]. Available: https://www.owasp.org/index.php/ASVS. [Accessed: 04-Dec-2014].

Appendix A: Implementation hints

During the implementation phase, two types of difficulties were faced as the following:

1. **Infrastructure setup**: Configuration of different systems and make them work is difficult sometimes do to bugs or lack of features. For example it turned out that installing a CA Server and a PKI infrastructure needs hand-on experience and expertise.
2. **Coding skills**: Since the secure system uses different ciphers and algorithms, finding proper coding libraries and proper usage of them is extremely difficult and time consuming. It is to be advised to do such projects alongside another person who has extensive coding skills.

Some hints for infrastructure implementation:

1. The whole infrastructure can be implemented based on open source hosting operating system such as Linux. However due to ease of installation and configuration Windows 2008 Server Standard Edition has been selected. The institute has an agreement called "Microsoft Dreamspark Premium" which gives a free license for the Windows installation.
2. Windows Server 2008 has strict rules for its firewall and also user access control to avoid unauthorized installations on the server. Make sure to change the firewall rules, so it does not block the ports that the IM server is listening to which are port number 5222 for TLS and 5223 for old SSL.
3. If you want to test the network connectivity and reachability of the hosting server via "Ping" command, be sure that to change the rules of firewall to allow this. Otherwise you are not be able to ping the server and this might misguide you.
4. By default, Openfire IM server does not archive chat conversations on the server. Enable this feature from the administration panel of the server. You can also specify how many days, the chat conversations remain on the server.
5. If you want to test the IM server after installation, the creator of Openfire, has an application called "Spark"[*] which can be downloaded and installed on a normal laptop and the server. Basically, Spark is an IM client for PCs available for different operating systems. Now as the admin of the IM server, you can easily create two users and login with them on both systems and start to chat and to see if you can send and receive messages in both clients' applications. After this success you are good to go for coding the mobile application and to test it.

[*] http://www.igniterealtime.org/downloads/index.jsp#spark