SECURE COMMUNICATION THROUGH UNTRUSTED SERVER

By

Labony Sarkar

MASTER OF ENGINEERING

IN

INFORMATION AND COMMUNICATIN TECHNOLOGY

INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

2017

The project titled "**SECURE COMMUNICATION THROUGH UNTRUSTED SERVER**" submitted by Labony Sarkar, Roll No.: 0411312011, Session: April 2011, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Master of Engineering in Information and Communication Technology on March 29, 2017.

BOARD OF EXAMINERS

1. **Dr. Hossen Asiful Mustafa** — Chairman
   Assistant Professor
   IICT, BUET, Dhaka

2. **Dr. Md. Saiful Islam** — Member
   Director and Professor
   IICT, BUET, Dhaka

3. **Shahin Akhter** — Member
   Lecturer
   IICT, BUET, Dhaka

CANDIDATE'S DECLARATION

It is hereby declared that this project or any part of it has not been submitted elsewhere for the award of any degree or diploma.

_____

Labony Sarkar

DEDICATION

I dedicate this project to God Almighty my creator, my strong pillar, my source of inspiration, wisdom, knowledge and understanding. He has been the source of my strength throughout this program and on His wings only have I soared. I also dedicate this work to my husband who has encouraged me all the way and whose encouragement has made sure that I give it all it takes to finish that which I have started. I also dedicate this work to my family for their endless love, support and encouragement.

Table of Contents

List of Figures

List of Tables

List of Abbreviations of Technical Symbols and Terms

API	Application Program Interface
BTS	Base Transceiver Station
GPS	Global Positioning System
IM	Instant Messaging
JSON	JavaScript Object Notation
MAC	Media Access Control
MMS	Multimedia Messaging System
OS	Operating System
PDA	Personal Digital Assistant
PKI	Public Key Infrastructure
POS	Point of Sale
SDK	Software Development Kit
SMS	Short Messaging System
TCP	Transmission Control Protocol

Acknowledgement

First of all, I would like to express my gratefulness to God who offers me his blessings to complete this project work. I would like to thank my project supervisor Dr. Hossen Asiful Mustafa, Assistant Professor, Institute of Information and Communication Technology, Bangladesh University of Engineering and Technology, for his proper guidance in selecting the research area and supporting by all means in the process of this project. His proper guidance and efforts made this work possible. I am highly grateful to my parents, family members, colleagues, friends and senior fellows who are always helped me and gave me the moral support and inspiration. Last, but not the least, I would like to thank the honorable director of IICT, BUET, Prof. Dr. Md. Saiful Islam and my Institution, Institute of Information and Communication Technology, Bangladesh University of Engineering and Technology, for giving me the opportunity for this research work and facilitate me throughout the whole Master's program.

Abstract

In recent times the usage of Smartphone has significantly increased. Instant Messaging (IM) is widely being used in smart phones. IM is a type of communication service over the Internet that enables individuals to exchange text messages and track availability of a list of users in real-time. Despite these good features and the continuous increase of Smartphone users, there have been very little concerns drawn towards the security measures of IM applications. For these kinds of applications, security is the main concern. These applications rely on servers to store user related information, key generation, key management, key exchange process and also the encryption of confidential information. Unfortunately, there always is a chance of malicious attacks in the server side. Anyone with access to the server may obtain all of the data stored there.

In most of the existing applications, the public and private key pair is generated at server-side and the private key is shared with the user for use in communication. A man-in-the-middle attack can compromise the private key during the sharing and thus, make the communication vulnerable. Existing systems do not provide the entire security of private key that should never be shared with someone other than the authorized entity.

The motivation for this work is the need to identifying security services of an IM application and to design a secure system for IM applications. Our prime goal is to restrict server to have access or store the private keys and ensure that the server can't read messages users are sending. We aim to provide cryptographically strong security while generating the public and private keys and exchanging them with the client. To ensure that only the intended parties get access to the sent data, asymmetric encryption is used. Every client has their own private keys stored client-side and their public keys stored on the server attached to the user in a database. This forces the private key to be safe from the malicious attacks in the server. In this way, we ensure secure communication through untrusted server.

CHAPTER 1
Introduction

In recent times the usage of smart phones has significantly increased. As a consequence, there is an increasing demand to have more and more mobile applications. A significant progress has already been made in mobile applications development. Instant Messaging (IM) is a type of communications service over the Internet that enables individuals to exchange text messages and track availability of a list of users in real-time. Messaging applications are one of the most popular applications for Smartphone. Recently, Smartphone IM application like Whatsapp, Facebook chat, Google Hangout and many more has taken over the Short Messaging System (SMS) and Multimedia Messaging System (MMS) in the capability of sending messages of various sizes [1]. The growing popularity of IM applications may also be associated with sending messages or initiating voice calls via Internet which makes it almost free of cost for the users to communicate with each other. Despite these good features and the continuous increase of Smartphone users, there have been very little concerns drawn towards the security measures of IM applications. Because of this observation, attackers are beginning to take interest in this channel to extract personal information of IM application users to be used for impersonation and other fraudulent activities [2].

In order to protect mobile applications and data being transferred from these applications, a number of security measures are taken in advance. Usage of cryptography in such mobile applications is one of these measures. Cryptography is one of the essential techniques to maintain confidentiality in communication. However, since all these mechanisms are done on server side, there always is a chance to malicious attacks on the server side. The clients have no options other than trust the server.

1.1 Problem Statement

Security of an application heavily depends on cryptography and the heart of cryptography is the keys used to encrypt/decrypt the data. A compromised key could

lead to the failure of the whole security system. In message communication system, so far application servers are providing security between two sender and receiver. The involvement of the server cannot ensure absolute security. Unfortunately, there always is a chance of attacks in the server side. A malicious server can access and read the content of any communication. Anyone with access to the server may obtain all of the data stored there. In most of the existing applications, the public and private key pair is generated at server-side and the private key is shared with the user for use in communication. A man-in-the-middle attack can compromise the private key during the sharing and thus, make the communication vulnerable.

Existing systems do not provide the entire security of private key that should never be shared with someone other than the authorized entity. This is a key requirement for the public key paradigm to work. In this project, we will implement a system which will ensure security of the private key without using the server and use the key for secure communication through untrusted server.

1.2 Motivation

The motivation for this project is the need to identifying secure key management system of an IM application and to design a cryptographic key management system that must restrict server to have access or store the private keys and server also should not know what messages users are sending. To achieve this during registration, a public-private key pair is generated in client side and only the public key along with user information is sent from a client to the server. When sender send message to receiver, first a public key request for receiver is sent to the server, after receiving the public key of the receiver, the message will be encrypted with receiver's public key and encrypted message will be sent to the server. Server then forwards the message to the receiver. Receiver will decrypt the message with own private key. Since the server does not have access to the private keys, it cannot read the communication.

1.3 Objectives

The objective of this work is to implement a system that can be used for secure communication through untrusted server. To achieve this objective, the project has set the following goals:

- To implement a secure key generation at client side.
- To implement a secure communication between Client and Server.
- To restrict server to have access or store the private keys and server also should not know what messages users are sending.
- To analyze the security strength in comparison of existing systems

The development and analysis of the communication system will involve a number of considerations as explained here briefly. We will develop a communication protocol which will use SSL/TLS to ensure secure communication with server and RSA, asymmetric key cryptography, to generate the public and private keys, to encrypt the message using the public key, to decrypt the message using the private key. We will develop an Android app along with server-side module as a case study. Finally, we will analyze the security of our system in comparison with the existing systems.

1.4 Organization of this project

This project consists of six chapters. The chapters are described as follows:

Chapter 1 introduces brief prospect of Smartphone messaging applications and discuss about coexistence security issues. After that, the motivation and objectives are also presented in this chapter.

Chapter 2 of this project work covers a detail overview of mobile phones evolutions and different mobile phone operating systems. Also describe different type of mobile services and the security issues for the applications.

Chapter 3 describes the cryptographic systems and their key management system. Also describe the importance of cryptographic keys and their security issues.

Chapter 4 presents RSA public key cryptography. It describes RSA key generation algorithm, encryption and decryption process in details. It also mentioned the security attacks against RSA.

Chapter 5 gives an overview of the proposed messaging application. This chapter also mentioned the implementation requirements and describes the implementation procedures. After that gives demo presentation of the application step by step and analysis the key generation system.

Chapter 6 concludes the project work and recommendations for future work is also available in this chapter.

1.5 Summary

This chapter introduces a very brief introduction of Instant Messaging applications with their various security issues. Detail descriptions of existing security issues with messaging applications are included here. Motivations that enticed us to conduct this project are also keeping in focus here. Finally, the objectives to propose new secure key generation system for ensure secure communication is mentioned in this chapter.

CHAPTER 2

Overview of Technology and Security

2.1 Mobile Phones

A mobile phone is a portable device which allows us to make and receive calls over a radio frequency link from a wide geographic area. Mobile phones were first introduced in 1973. Since then, a significant progress has been made in the development of mobile phones. As a result of this development, today's mobile phones are much more efficient in terms of processing power, storage and functionality. Modern phones don't only allow us to call, but we can also send SMS, MMS, browse Internet, send and receive e-mails, etc. [3]. The evolutions of Mobile Phones are shows in the following figure 2.1.

Figure 2.1 Evolution of Mobile Phones [4]

2.1.1 Smartphone

A mobile phone built with a mobile operating system is referred as "Smartphone". A Smartphone is far more capable of faster processing and connectivity. A Smartphone combines the functionalities of PDAs, media player, video and photographic camera, GPS, etc. Today's smart phones are manufactured with a high resolution touch screens. Using touch screen, we can interact with the phone to perform any action. Broadband and high speed Wi-Fi connectivity of Smart phones allows us to access

Internet and web pages. The following figure 2.2 gives an overview of some Smartphone and their processing speed is discussed in [3].

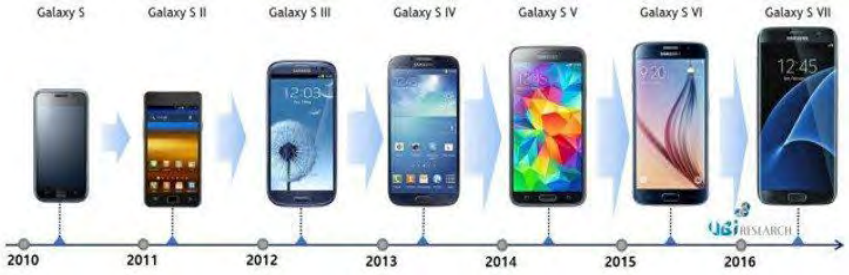

Figure 2.2 Smart phone up gradation up to 2016 [5]

2.1.2 Mobile Operating System

A mobile operating system is an operating system which runs on a smart phone and combines the functionality of mobile phone and personal computers. A mobile OS allows programmers to develop applications without considering specifications, drivers and functionality of hardware. Mobile OS allows these applications to customize hardware according to their design.

There are many Mobile OS available. The following are some popular Mobile OS.

2.1.2.1 Android

Android [6] is Linux kernel based mobile operating system developed by Google. It is an open source Mobile OS and can be customized according to phone's requirements. Third party applications can be installed on it. Android applications are developed using the Android software development kit (SDK) in Java programming language and it has complete access to the android APIs [7]. Updates for the open source Android mobile operating system have been developed under "dessert-inspired" version names (Cupcake, Donut, Eclair, Gingerbread, Honeycomb, Ice Cream Sandwich) with each new version arriving in alphabetical order with new enhancements and improvements. Many big Smartphone manufactures such as: Samsung, HTC, Sony, etc., are using Android with their smartphones. By 2016,

Android is majority shareholder in the market [9]. As of October 2016 the current version of android is 7.1 “Nougat”.

2.1.2.2 iOS

iOS [8] is the second most popular mobile operating system in global market [12]. iOS is a XNU kernel based Mobile OS which runs on iPhones, iPods and iPod touch. This mobile OS runs only on Apple device and allows applications to be installed in it, but only Apple’s approved applications can be installed in it. All applications for iOS can only be installed from Apple’s “App Store”. As of June 2016 App Store contained more than 2,000,000 iOS Applications. The programming language for iOS application is Objective C and it provides a rich set of APIs for building application. iOS is considered as one of the most mature and robust Mobile OS. As of September 2016 the current version of iOS is 10.

2.1.2.3 Windows Phone

Windows Phone [9] is mobile OS from Microsoft developed for Smartphone. It is a replacement successor of Windows Mobile. Although it is a Microsoft product, other Smartphone manufactures can obtain its license to use with their devices. It also allows mobile applications to run on it. Mobile applications for windows phone are developed using C Sharp (C#) and Microsoft Visual Studio. It also provides rich set of APIs and access to Smartphone resources. Windows Phone is relatively new Mobile OS and not very popular until now.

2.1.2.4 BlackBerry 10

BlackBerry 10 [10] is proprietary mobile OS from BlackBerry Limited developed for BlackBerry Smartphones and tablets. BlackBerry is not a very popular mobile OS. Therefore, recently BlackBerry has added an Android runtime layer. This would allow developers to easily package and distribute applications designed to work with Android.

The market share of different Mobile OS is shown by the following figure 2.3:

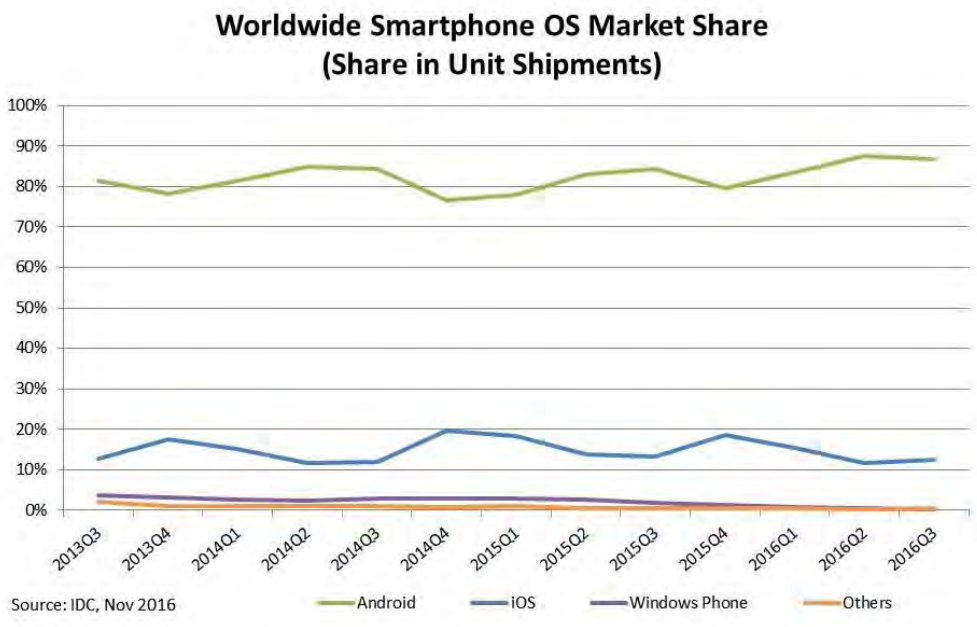

Figure 2.3 Smartphone sales with respect to OS [12]

2.1.3 Mobile Services

With the development of smart phone technologies, mobile applications have also grown-up speedily. Nowadays, lots of useful features can be used with mobile phones.

2.1.3.1 Instant Messaging (IM)

Instant Messaging (IM) is one of the most popular applications. Recently, Smartphone IM application like Whatsapp, Facebook chat, Google hangout and many more has taken over the SMS and MMS in the capability of sending messages of various sizes. It is a real-time communication. People of any part of the world need not to pay the international or domestic long distance charges associated with using the phone. Nowadays, many companies used IM as a part of their daily business routine. These applications have been successfully acquire millions of users.

2.1.3.2 Mobile Payments

Mobile Payment is a payment system which operates under financial regulation and uses a mobile to make money transactions. Instead of paying with cash, cheque, or credit cards, a consumer can use a mobile phone to pay for a wide range of services and digital or hard goods.

Mobile payment is a transfer of funds in return for goods or services in which a mobile device is functionally involved in executing and confirming payment. The payer can be standing at a POS or be interacting with a merchant located somewhere else. Consumers can use a mobile device to pay for goods and services such as: Music, videos, ringtones, online game subscriptions, wallpapers, and other digital goods, transportation-related items, such as bus, subway, or train fares and parking at meters and any merchandise in a physical merchant location [14].

2.1.3.3 Mobile POS

Mobile POS is a system developed to use mobile devices to replace the traditional merchant POS of sale terminal and is typically used for inventory management and electronic payments. Mobile POS can support different types of payment devices, such as traditional stripe card, contactless bank cards, etc. [14].

2.2 Security Requirements

The growing popularity of IM applications may also be associated with some of the benefits it brings to the users. For example, users being able to see each other via webcams or talk directly for free over the internet using a microphone. IM applications have features like free audio and video calls, sending multimedia messages of various sizes; sharing data between contacts is made easy with user friendly options, and the availability for credit card usage amongst others. Despite these good features and the continuous increase of Smartphone users, there have been very little concerns drawn towards the security measures of IM applications. Because of this observation, attackers are beginning to take interest in this channel to extract personal information of IM application users to be used for impersonation and other fraudulent activities [15] [16].

2.2.1 Security Threats

Understanding the basics of message security opens the door to preventing some common security threats in message usage.

- **Man-in-middle Attack:** This is the network that authenticates users. The user does not authenticate network so the attacker can use a false BTS with the same mobile network code as the subscriber's legitimate network to impersonate himself and perform a man-in-the-middle attack.
- **Replay Attack:** The attacker can misuse the previously exchanged messages between the subscriber and network in order to perform the replay attacks.
- **Message Disclosure:** Since encryption is not applied to message transmission by default, messages could be intercepted and snooped during transmission. In addition, messages are stored as plain text by the mobile networks before they are successfully delivered to the intended recipient. These messages could be viewed by users in the network who have access to the messaging system.
- **Spamming:** While using message as a legitimate marketing channel, many people have had the inconvenience of receiving message spam. The availability of bulk message broadcasting utilities makes it easy for virtually everyone to send out mass messages.
- **Denial of Service (DoS) Attacks:** DoS attacks are made possible by sending repeated messages to a target mobile phone, making the victim's mobile phone inaccessible.
- **Phone Crashes:** Some vulnerable mobile phones may crash if they receive a particular type of malformed short message. Once a malformed message is received, the infected phone becomes inoperable.
- **Message Viruses:** There have been no reports of viruses being attached to short messages, but as mobile phones are getting more powerful and programmable; the potential of viruses being spread through message is becoming great.

- **Message Phishing:** Message phishing is a combination of message and phishing. Similar to an Internet phishing attack using email, attackers are attempting to fool mobile phone users with bogus text messages.

We can gather more details knowledge about the above security threats from reference book [18].

2.2.2 Security Requirements for IM Services

Security Requirements can be referred as the requirements which need to be fulfilled in order to declare a system secure. These requirements include the following:

2.2.2.1 Confidentiality

Confidentiality means the content of message/data is protected in such a way that it can only be viewed by authenticated entity. Data/message must be protected in such a way that no interceptor can intercept and get the context of original data/message. Confidentiality can be achieved by cryptography. When talking about financial data, it is really very important to protect the data.

2.2.2.2 Integrity

Integrity means that data/message is not tampered during transfer between two communicating parties. An attacker can attack data/message during the transfer and could modify it to benefit him/herself.

2.2.2.3 Authentication

This is verification of the identity of a user or a system before granting it the access to protected information or system. Without a well-defined authentication system an attacker can attack the system and violate security of the system. A well-defined and strong authentication system can guard against such attacks and ensure the safety of a system.

2.2.2.4 Non-Repudiation

Non-repudiation provides the integrity of the source of the data/message. This means a source of the message cannot deny its involvement in some action of sending and receiving message. Sometimes, attackers can send a wrong message to an entity

acting as its communication partner; however they are not real partners. The purpose of this attack is to accuse the party involved in a communication. Non-repudiation can be achieved by digital signatures.

The above security requirements are discussed in detail in [17-19].

2.3 Summary

Smart phones are amazing tools for staying connected with friends and family, for keeping up to the minute with work, for shopping, for paying bills and managing free time. Cybercriminals are seeing an opportunity to prey on victims by attacking mobile devices, driving an increase in the number and sophistication of threats. For this reason, more diligence and awareness is required for the end users.

CHAPTER 3

Cryptography and Cryptographic Key

3.1 Cryptography

Cryptography is an art of secret communication where a plain text is converted into text non-understandable for third party. The process of this conversion is known as encryption. The encryption is performed with the help of some key known as cryptographic key, also sometimes as encryption key. At the other end of communication, the receiver receives the encrypted text and decrypts that text. The receiver uses cryptographic key / decryption key to convert the text back to understandable form. Cryptography is done to achieve confidentiality.

There are two main approaches to cryptography:

- Asymmetric Cryptography
- Symmetric Cryptography

3.1.1 Asymmetric Cryptography

Also known as public/private key cryptography, this is an approach of cryptography in which a pair of keys is used for encryption and decryption process [17-18]. Figure 3.1 shows the process of asymmetric key cryptography. There are two keys in this pair, they both are different from each other, but they are associated with each other. These two keys are:

- Public Key
- Private Key

Public Key is a key which is used to encrypt plain text. As we can see by its name, public key means, this key is known to public or to everyone in communication. Anyone can use this key to encrypt data using this key before sending it to its owner.

Private Key is used to decrypt data back to its plain form. This is a secret key and only known to its owner. If private key in asymmetric encryption is compromised, the whole system's security will be at risk.

Asymmetric encryption can provide a better security solution, but it requires more computational power and resources which means a decrease in systems performance.

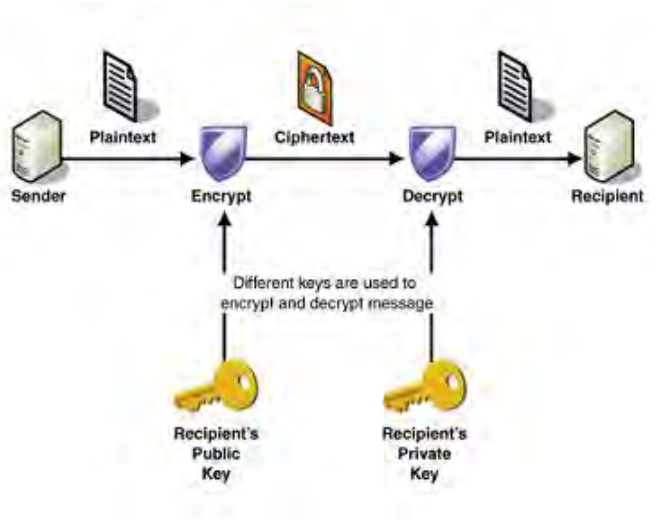

Figure 3.1 Asymmetric Key Cryptography

One of the major drawbacks of asymmetric cryptography is the distribution of key to legitimate parties. Public Key Infrastructure (PKI) is a viable solution for wired network, but small scale devices still lack suitable computational power for handling resource consumptions.

3.1.2 Symmetric Cryptography

Also known as secret key cryptography or shared key cryptography, symmetric cryptography is an approach in which a single secret key is used for both encryption and decryption [17-18]. Experts recommend the use of symmetric cryptography technique to encrypt messages, while public key cryptography should be used for session protection. Figure 3.2 shows the process of symmetric key cryptography.

Since in shared key cryptography one shared key is used, it is very important to protect the shared key. A compromised key could be a threat to confidentiality of data. For protection of shared key, a well design key management system has a great importance.

Symmetric encryption is mostly done to achieve confidentiality of data. Encrypted data could be sent over insecure channel. However, the key must be sent through a different channel.

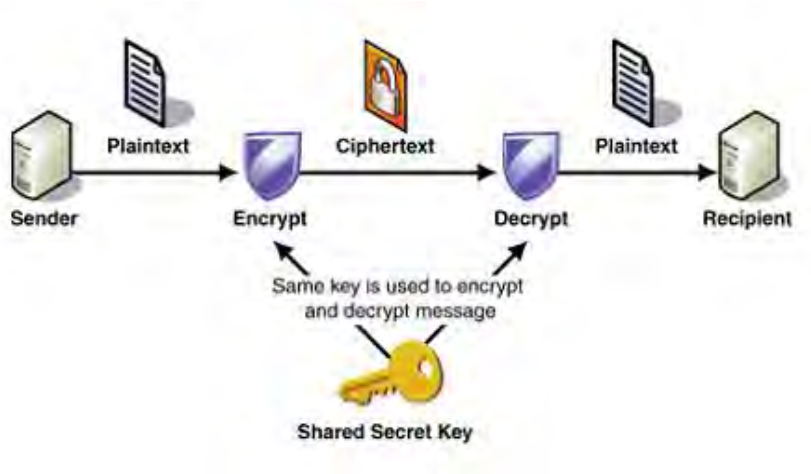

Figure 3.2 Symmetric Key Cryptography

More details of symmetric key cryptography are available from references book [13][15].

3.2 Importance of Keys in Cryptography

When talking about information security, cryptography is considered as heart of confidentiality. Confidentiality heavily depends on cryptography. A well-chosen cryptographic technique, i.e. asymmetric cryptography or shared key cryptography, could be vital to achieve the required level of confidentiality. Keys in cryptography have great importance or could be considered as the most important part of cryptography. Any compromised key could lead to failure of the entire security system. We can choose different key sizes to increase the protection level in cryptography but it is also very important to secure the key itself. A secret key must be kept secret and should be stored securely, so that no adversary can access it. A well designed key management system could help protecting the keys in an effective way [11][17][20].

3.3 Key Management

Key Management is a system of managing keys for a cryptosystem. Key management process includes generation, storage, exchange, use and replacement of keys. A successful and secure key management system is critical for the security of every cryptosystem.

3.3.1 Key Management Functions

3.3.1.1 Key Generation

A key management system must generate random keys in a way that it is not feasible to determine the next random number.

3.3.1.2 Key Storage

A key management system must provide the facility to store the key within its boundaries in such a way that the keys can be accessed and used whenever needed.

3.3.1.3 Key Exchange

The key management system must allow the exchange of keys among parties involved in the communication.

3.3.1.4 Key Replacement

Key management system must provide the functionality of key replacement in case of compromised keys or whenever an update is needed according to security policy.

3.3.2 Security Requirements for Key Management System

A secure and efficient key management is the goal of every system. Protection measures must be taken to secure the information within a key management system.

3.3.2.1 Confidentiality

When keying material in transmitted over an insecure channel, such as Internet, it must be protected in a way that no adversary can understand it. This could be achieved by encrypting the information before transmitting it to an insecure channel. Or it should be transferred using some secure channel.

3.3.2.2 Integrity

The keying material could be at risk of alteration during the communication. Therefore, a key management system should have the ability to both detect and prevent the information from being altered. This can be achieved by cryptographic mechanisms, such as MAC or digital signature.

3.3.2.3 Availability

The keying information could be destroyed intentionally or unintentionally during communication between two parties. It is important to take necessary measures to make keying information available at its destination. This cannot be achieved by cryptography, so some non-cryptographic mechanisms should also be part of key management system to ensure availability.

3.3.2.4 Information in Storage

The cryptographic information which is not in transit and is stored in some sort of storage device, i.e. hard disk or any other storage medium, is also at risk of being compromised. Therefore, such information must also be protected. The following are some key requirements to ensure the security of cryptographic information located at storage devices.

3.3.2.5 Confidentiality

Private or secret keys must be protected in a way that no one can view them when they are persisted in storage. This can be achieved by restricting access to storage medium. Also private/secret keys must be stored in encrypted form.

3.3.2.6 Integrity

Stored cryptographic information is always at high risk of modification. Therefore, physical or cryptographic measures must be taken to ensure the integrity of the information. Physically it can be achieved by restricting the access to cryptographic material. Cryptographic mechanisms, such as: MAC or digital signature could also be used to ensure the integrity of cryptographic information. Compromised information could be restored from secure backup of information.

3.3.2.7 Availability

The cryptographic information must be available when needed. Sometimes it is possible to lose the information stored in the storage device. So, a secure and encrypted backup of the information must be taken and securely stored on separate storage devices.

3.3.3 Key Security Policy

It is very important to protect the key from being compromised. A compromised key could be disastrous for the entire system. Key security can be achieved by a well-defined key security policy. A well-defined key security policy is very important for an effective and reliable key management system. An effective key security policy must define the complete life cycle of a key which includes key generation, storage, distribution, expiration and updating under a protective environment. More details of cryptographic keys are available from references [7][13][15-17].

3.4 Summary

Cryptography is the foundation of protecting electronic data and cyber security. Encryption can effectively prevent breaches while also protecting both consumer privacy and sensitive data. With effective cryptographic key management, data that is encrypted can still be protected even in the event of a breach, since encrypted data cannot be decrypted without the right keys. Proper usage of encryption and key management will assist an organization's efforts to protect their data. To be effective, encryption requires confidential and strong keys to protect against breaches. Data protected by cryptography is dependent upon the strength of the keys along with the mechanisms and protocols used in association with those keys. Secret and private keys must be held securely without unauthorized disclosure and protected from modifications. During their entire life cycle, keys must be managed with proper procedures and protocols to ensure they are not compromised. From the moment of their generation and throughout distribution, storage, entry, use, destruction and archival, keys must be handled with established protocols.

CHAPTER 4

PUBLIC KEY CRYPTOGRAPHY

4.1 Introduction

The RSA public key cryptosystem was invented by R. Rivest, A. Shamir and L. Adleman [17-18]. The RSA cryptosystem is based on the dramatic difference between the ease of finding large primes and the difficulty of factoring the product of two large prime numbers.

- RSA is a public key algorithm invented by Rivest, Shamir and Adleman. The key used for encryption is different from (but related to) the key used for decryption.
- The algorithm is based on modular exponentiation. Numbers e, d and N are chosen with the property that if A is a number less than N, then $(A^e \ mod \ N)^d \ mod \ N \ = A$
- This means that you can encrypt A with e and decrypt using d. Conversely you can encrypt using d and decrypt using e (though doing it this way round is usually referred to as signing and verification).
    - The pair of numbers (e, N) is known as the public key and can be published.
    - The pair of numbers (d, N) is known as the private key and must be kept secret.
- The number e is known as the public exponent, the number d is known as the private exponent, and N is known as the modulus. When talking of key lengths in connection with RSA, what is meant is the modulus length.
- An algorithm that uses different keys for encryption and decryption is said to be asymmetric.
- Anybody knowing the public key can use it to create encrypted messages, but only the owner of the secret key can decrypt them.
- Conversely the owner of the secret key can encrypt messages that can be decrypted by anybody with the public key. Anybody successfully decrypting

such messages can be sure that only the owner of the secret key could have encrypted them. This fact is the basis of the digital signature technique.

More details of cryptographic keys are available from references [13][15][18-19].

4.2 RSA Encryption

Suppose Bob wishes to send a message (say 'm') to Alice. To encrypt the message using the RSA encryption scheme, Bob must obtain Alice's public key pair (e, n). The message to send must now be encrypted using this pair (e, n) . However, the message 'm' must be represented as an integer in the interval $[0, n-1]$ To encrypt it, Bob simply computes the number 'c' where $c = m^e \ mod \ n$. Bob sends the cipher text c to Alice.

4.3 RSA Decryption

To decrypt the cipher text c, Alice needs to use her own private key d (the decryption exponent) and the modulus n. Simply computing the value of $c^d \ mod \ n$ yields back the decrypted message (m).

4.4 Attacks against RSA

Through the basic algorithm is secure, there are attacks on how RSA is implemented [16-17]

1. Forward search attack: If message space is predictable, attacker can decrypt C simply by encrypting all possible messages until a match with C is obtained.
2. Common modulus attack: If everyone is given the same modulus n but different (e, d) pair, then under certain conditions, it is possible to decrypt the message without d.
3. Low encryption exponents: When encrypting with low encryption exponents (e.g., $e = 3$) and small values of the m, (i.e. $m < n_1/e$) the result of m e is strictly less than the modulus n. In this case, cipher texts can be easily decrypted by taking the e_{th} root of the cipher text over the integers.
4. RSA has the property that the product of two cipher texts is equal to the encryption of the product of the respective plaintexts. That is $m_1^e m_2^e =$

$(m_1 m_2)^e (mod\ n)$ Because of this multiplicative property a chosen-cipher text attack is possible.

4.5 Key Generation Algorithm

1. Generate two large random primes, p and q, of approximately equal size such that their product $n = pq$ is of the required bit length, e.g. 1024 bits.
2. Compute $n = pq$ and $\varphi = (p-1)(q-1)$.
3. Choose an integer e, $1 < e < \varphi$, such that $gcd(e, \varphi) = 1$.
4. Compute the secret exponent d, $1 < d < \varphi$, such that $ed \equiv 1 (mod\ \varphi)$.
5. The public key is (n, e) and the private key (d, p, q). Keep all the values d, p, q and φ secret. [We prefer sometimes to write the private key as (n, d) because you need the value of n when using d.]
    - n is known as the modulus.
    - e is known as the public exponent or encryption exponent or just the exponent.
    - d is known as the secret exponent or decryption exponent

4.6 RSA ALGORITHM

1. Choose two distinct prime numbers, p and q.
2. Let $n = pq$.
3. Let $\varphi(pq) = (p-1)(q-1)$. (φ is totient function).
4. Pick an integer e such that $1 < e < \varphi(pq)$, and e and $\varphi(pq)$ share no divisors other than 1 (e and $\varphi(pq)$ are co-prime).
5. Find d which satisfies.

   d is a secret private key exponent.

1. The public key consists of e (often called public exponent) and n (often called modulus). The private key consists of e and d (private exponent).

The message m is encrypted using formula. Where c is the encrypted message. The encrypted message is decrypted using formula.

Encryption and decryption formulas show how to encode and decode a single integer. Bigger (or different) pieces of information are encoded by converting them

into (potentially large) integers first. As RSA is not particularly fast, it is usually only to encrypts the key of some faster algorithm. After RSA decrypts the key, this supplementary algorithm uses it to decrypt the rest of the message.

RSA algorithm is fundamentally based on the Euler theorem:
Where a and n are positive integers and a is a co-prime to n. To break the algorithm from the mathematical side, one needs to solve the factoring problem (find the two prime numbers that, when multiplied, produce the given result). When the picked numbers are large enough, the problem cannot be easily solved by brute force and at least currently it also does not have easier analytic solution.

4.7 Encryption Algorithm

Sender A does the following:-

1. Obtains the recipient B's public key (n, e).
2. Represents the plaintext message as a positive integer, $1 < m < n$.
3. Computes the cipher text $c = m^e \ mod \ n$.
4. Sends the cipher text c to B.

4.8 Decryption Algorithm

Alice can recover m from c using her private key d in the following procedure:

$$M = c^d mod \ n$$

Given m, she can recover the original message M.

More details of cryptographic keys are available from references [13-21].

4.9 Summary

RSA is the most popular public-key cryptosystem available today. RSA is incorporate into all of the major protocols for secure Internet communications. Its popularity stems from the fact that it can be used from both encryption and authentication, and that it has been around for many years and has successfully withstood much security. The security of RSA is related to the assumption that factoring is difficult.

CHAPTER 5

System Design and Implementation

5.1 Overview of the System

The proposed messaging system is a client-server based system. This system allows the application to connect to the server and send messages. This system includes a registration step during which key pair is generated and only the public key is send to the server along with user information. It has a Login step and after successful login one can start messaging.

5.1.1 Registration

All users of the system need to register to application. After installing the app to user mobile, user should register with required information. User will input user name, user ID and password and click to the Register button. During registration process application will generate the public and private key pair for the user. This key pair will be used for message encryption and decryption purpose. After key generation, client app will store the keys in two separate text files. Only public key including user information will send to the server for registration request. Server will store the user information and send a success response to the client. User will return to the login screen. The process is summarized in Tables 5.1 and shown in Figure 5.1.

Sl No	Actor	Description
1	User	1. Input User name, User ID and Password 2. Press 'Register' button
2	Client App	1. Generate Public and Private Key pair. 2. Store key pair in client's mobile memory in two separate text files. 3. Send only Public key with User information to the server.
3	Server	1. Store user's public key along with user information. 2. Send Registration successful response to the client.

Table 5.1 Registration Process

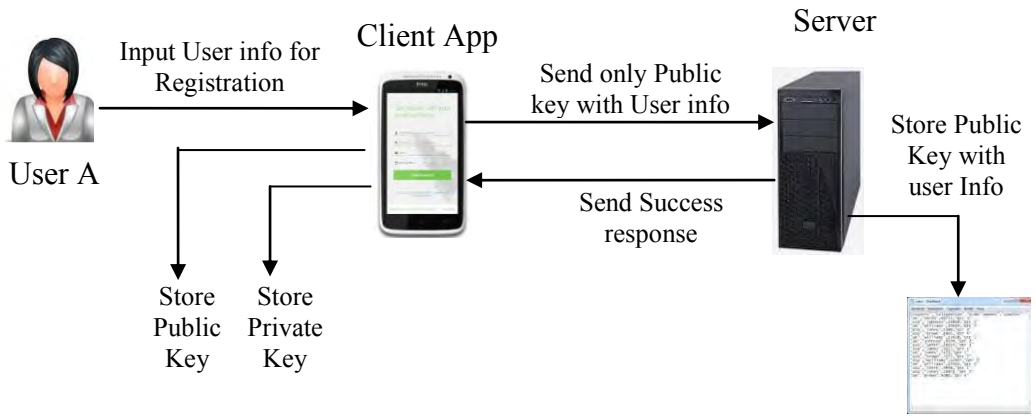

Figure 5.1 Registration Flow

5.1.2 Login

After Registration users can log into the system with user id and password. User put the user ID and password into the login form and press login button. Client app will sent Login request to server with user ID and password. Server will verify the user. If the user will be authenticated, then server will retrieve active user list and send to the logged user with login success response. User will get user list with a new page. Logged user can select any other user from the list and send message. The process is summarized in Tables 5.2 and shown in Figure 5.2.

Sl No	Actor	Description
1	User	1. Input User ID and Password 2. Press 'Login' button
2	Server	1. Search user information from stored data 2. If exists, Send Login successful response with active user list to the client.
3	Client App	1. After getting success response from the server, go to message page.

Table 5.2 Login Process

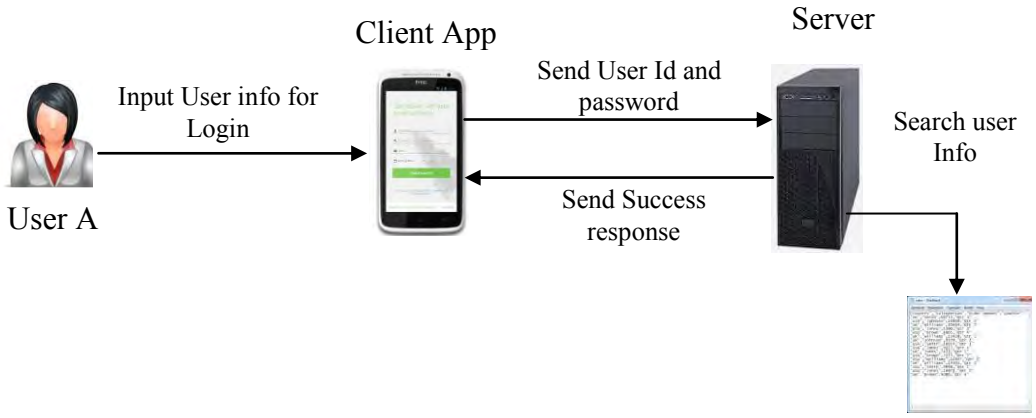

Figure 5.2 Login Flow

5.1.3 Send Receive Message

After successful login any user can send and receive message from another active user. For sending message, logged user will need to select a receiver from the list and write the text and click send button. Client app will send a public key request for receiver to the server. Server will search the public key matched with receiver and send to the client. Client will then encrypt the message with receiver's public key and send to the server. Server just forwards the message to the receiver. Receiver will decrypt the message by own private key. The process is summarized in Tables 5.3 and shown in Figure 5.3.

Sl No	Actor	Description
1	Sender A	1. Input message for receiver B. 2. Press 'Send' button.
2	Client App	1. Send a Public key request for B to the server.
3	Server	1. Search Public key of B in the file. 2. If exists, send to the Sender A.
4	Client App	1. After getting public key of B, encrypt message with B's Public key. 2. Send encrypted message to the server.
5	Server	1. Server just forward the encrypted message to the Receiver B.
6	Receiver B	1. After getting the encrypted message, B retrieves its private key from the file. 2. B decrypt the message with own Private Key and get the original message.

Table 5.3 Message Send and Receive Process

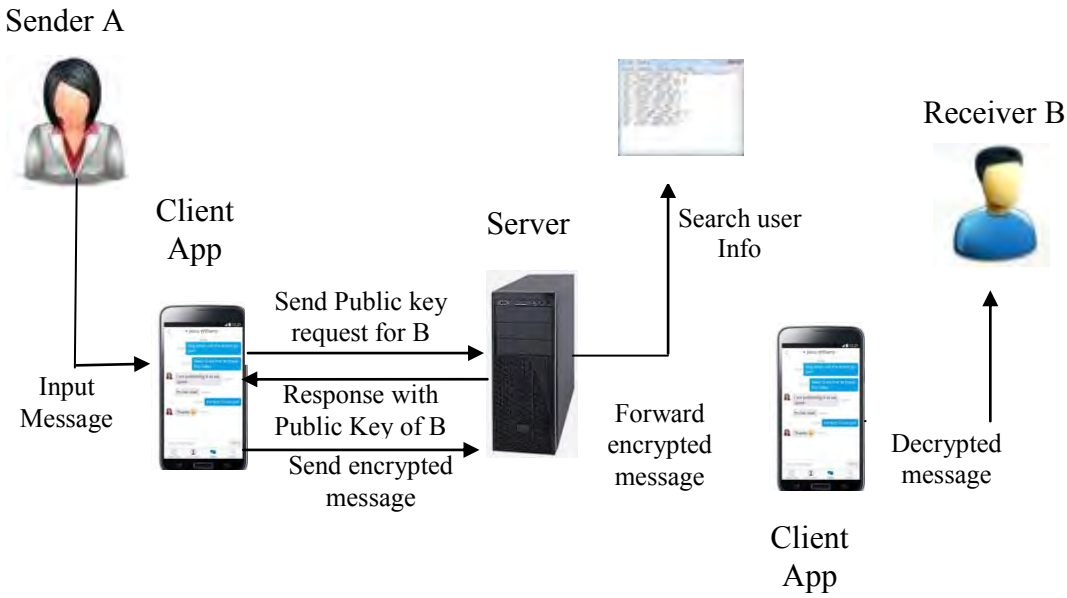

Figure 5.3 Message Send and Receive Flow

5.2 Implementation

5.2.1 Algorithm Selection

When talking about algorithm selection for asymmetric cryptography we have the possibility to choose from different asymmetric cryptography algorithms, such as RSA, DSA, and Elliptic Curve. However, we have chosen RSA as our Key generation and encryption decryption algorithm in this development process. RSA keys have a complex internal structure with specific mathematical properties. RSA allows us to choose different key sizes, such as 1024, 2048, 4096 bit. Larger keys provide more security, currently 1024 and below are considered breakable, and 2048 or 4096 are reasonable default key sizes for new keys. RSA encryption is interesting because encryption is performed using the public key, meaning anyone can encrypt data. The data is then decrypted using the private key.

5.2.2 Platform for Client

There are target platforms available such as iOS, Windows, BlackBerry but we have chosen Android as our target platform to design our instant messaging client application. Android is an open source and most popular Mobile OS and currently holds the biggest share in the market. By choosing Android we can cover a bigger target market. To implement security features we have used Java cryptographic extensions.

5.2.3 Development Environment

5.2.3.1 Java Technology

For the development of server side Instant Messaging application, we have used Java technology [28]. Java is widely used and provides a powerful set of libraries.

5.2.3.2 Eclipse IDE

We have used Eclipse IDE Kepler [29] for the development of both Server and Android Client. The development of server is done under standard Java environment. For Android client we have added Android SDK plug-in in Eclipse IDE.

5.2.3.3 Socket Server

For the exchange of requests and data between communication Server and Android client we have used simple Socket Server technology using java libraries.

5.2.3.4 Android SDK

To develop Android Client Application, we have used Android SDK [30] with Eclipse IDE. Android SDK includes powerful set of development tools and libraries.

5.2.3.5 JSON

JSON [31] is an open standard for exchange of data between exchange parties. It is easy and human readable format. We have used JSON (JavaScript Object Notation) data interchange format to interchange data between server and Android client.

5.3 Working Procedure

1. First we'll have a socket server running. When the android app connects to socket server, the server opens a **TCP connection between server and client**. The server is capable of opening concurrent connections when there are multiple clients.
2. When a socket connection is established few callback methods like **onOpen**, **onMessage**, **onExit** will be triggered on the both the ends (on server side and client side). There will be another method available to send message from client to server, or vice versa.

   onOpen() – This method is called when a new socket client connects.
   onMessage() – This method is called when a new message received from the client.

   onClose() – This method is called when a socket client disconnected from the server.

   sendMessageToClient() – This method is used to send a message to the clients.
3. JSON strings will be exchanged between server and client as a communication medium. Each JSON contains a node called flag to identify the purpose of JSON message. Below is example of JSON when a client Register the application that contains the client name, session id and number of people online.

```
{
  "message": "Registration",
  "flag": "Reg",
  "sessionId": "4",
  "userName": "Labony",
  "userId": "Labony",
  "password": "1234",
  "publicKey": "MIGfMA0GCSqGSEBAQUAQKBgQC",
  "onlineCount": 6
}
```

4. Whenever a JSON message received on client side, the JSON will be parsed and appropriate action will be taken. The following is a sample of JSON for message sending where ‘bbb’ send a message to ‘aaa’.

```
{
        "flag":"Msg",
        "sessionId":"2",
        "userId":"bbb",
        "message":"sj5TavjJ5ds9tIBemVq8fBeCPFqAoZsz4I",
        "receiverId":"aaa"
}
```

5.3.1 Client-Server Connection

A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to. A server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request. An endpoint is a combination of an IP address and a port number. Every TCP connection can be uniquely identified by its two endpoints. That way we can have multiple connections between our host and the server.

In our server side we do the followings:

- We use javax.websocket.* package in the Java platform which provides a class, Socket, that implements one side of a two-way connection between our server and android clients on the network.
- We set @ServerEndpoint("/chat").
- We deploy our server war in a real IP server and open default port 8080 listening for new client connection.
- To store all the live sessions we do

```
Set<Session> sessions = Collections.synchronizedSet(new HashSet<Session>());
```

- To decode the connection URL, we use java.net package, which has a class URLDecoder.
- To mapping between session and connection name we do

HashMap<String, String> nameSessionPair = new HashMap<String, String>();

- A method onOpen(Session session) is Called when a socket connection is opened. We use a variable to identify the user action name ‘name’. We decode the ‘name’ from the URL and do the appropriate action at server side.
- To send a message to the client, we use basic javax.websocket.Session.getBasicRemote().sendText() class.

In our client side we do the followings:

The client will know the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client tries to rendezvous with the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system.

- In our client side we define the connection URL in WsConfig.java class as like below

URL_WEBSOCKET = "ws://server_IP_adress:port/MyChatServer/chat?name=";

- In our client side, for connection handling we use a separate android project named android-websockets-master. We use this project as a jar in our main android application. We create websocket client as

```
client = new WebSocketClient(URI.create(WsConfig.URL_WEBSOCKET
                + URLEncoder.encode(name)), new WebSocketClient.Listener(){
 @Override
   public void onConnect() {
   Log.d(TAG, "Connected!");
   }
 @Override
   public void onMessage(String message) {
   Log.d(TAG, String.format("Got string message! %s", message));
```

```
    }
 @Override
    public void onMessage(byte[] data) {
    Log.d(TAG, String.format("Got binary message! %s", toHexString(data)));
    }
 @Override
    public void onDisconnect(int code, String reason) {
    Log.d(TAG, String.format("Disconnected! Code: %d Reason: %s", code, reason));
    }
 @Override
    public void onError(Exception error) {
       Log.e(TAG, "Error!", error);
    }
}, extraHeaders);
client.connect();
```

- With the ‘name’ variable we connect with the server.

Both client and server side we use some tag name to do a specific operation like ‘Reg’ for registration, ‘Log’ for login and ‘Msg’ for message.

5.3.2 Key Pair Generation

In our project, public private key pair is generated in registration process. For registration we create an activity name RegActivity.java. In this activity, we take input user information and generate public private key pair and finally make the connection string ‘name’ to connect with the server. To generate key pair we use java.security package and bouncy castle crypto API. The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms; it was developed by the Legion of the Bouncy Castle. The package is organized so that it contains a light-weight API suitable for use in any environment.

- Key pair generation we use the following code

```
//Object declaration
static KeyPairGenerator kpg;
```

```
static KeyPair kp;
static PublicKey publicKey;
static PrivateKey privateKey;
//Add provider
Security.addProvider(new org.bouncycastle.jce.provider.BouncyCastleProvider());
SecureRandom random = new SecureRandom();
kpg = KeyPairGenerator.getInstance("RSA","BC");
//Initialize key size
        kpg.initialize(1024,random);
//Generate Key Pair
        kp = kpg.genKeyPair();
// Get Public key
        publicKey = kp.getPublic();
//Convert Public key as string for storing in a text file
X509EncodedKeySpec x509EncodedKeySpecPub = new X509EncodedKeySpec(
                                                        publicKey.getEncoded());
publicK=new
            String(Base64.encodeBase64(x509EncodedKeySpecPub.getEncoded()));
//Save Public Key
ReadWriteKeys.writePublicKeyToFile(publicK);
// Get Private key
        privateKey = kp.getPrivate();
//Convert Private key as string for storing in a text file
PKCS8EncodedKeySpec keySpec = new
                                    PKCS8EncodedKeySpec(privateKey.getEncoded());
privateK=new String(Base64.encodeBase64(keySpec.getEncoded()));
//Save Private Key
ReadWriteKeys.writePrivateKeyToFile(privateK);
```

5.4 Demonstration

This section will show the complete working of the prototype of the developed Instant Messaging system using screen shots.

5.4.1 User Registration

When the client side application starts, the first step is user registration. It is necessary to register to start using the application. The client app generates public private key pair and key pair store in client's phone memory. The user information with public key will be sent to the server. Server will register the user and information with the public key will be stored in a file.

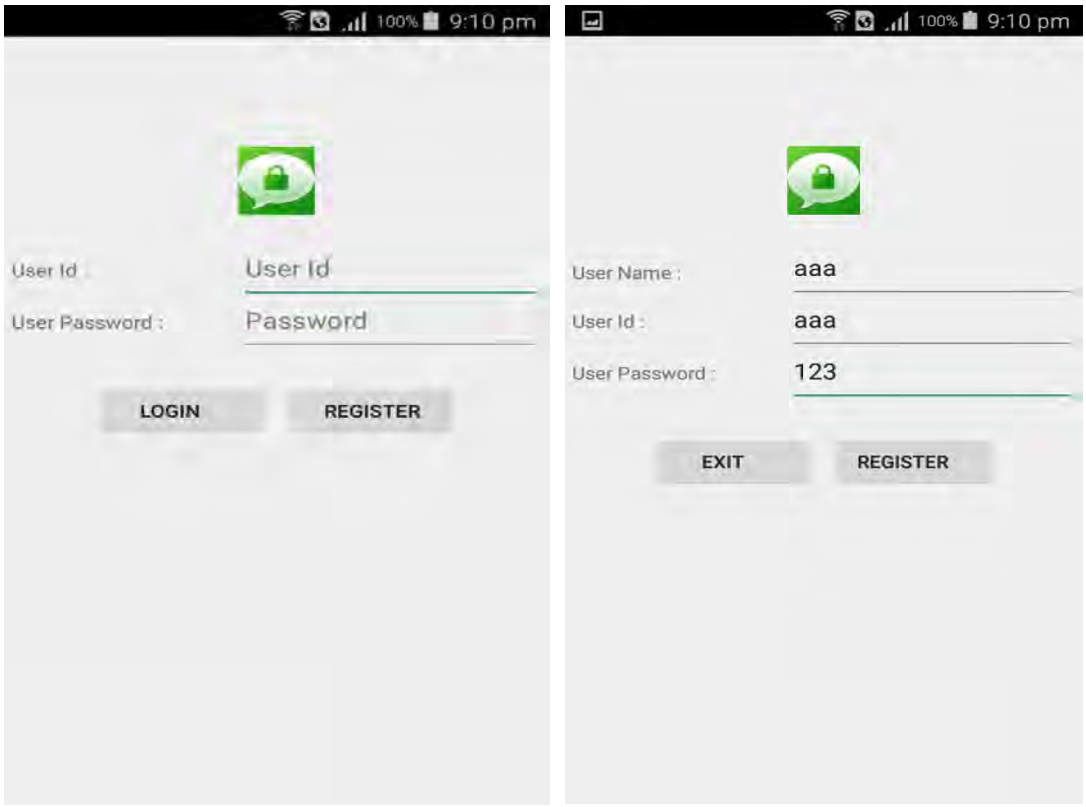

Figure 5.4 Screen shots of Registration data input

If the user is already registered it can simply login by providing its user id and password.

1. By clicking Register button, app will generate the key pair and the console output is shown below

I/System.out(29638): publicKey: RSA Public Key
I/System.out(29638): modulus:
f8270327ed0fc76eeae945c64c8372068459624613256b44e0142ac2c4718d16ab8684
95e033d137a3726fe3abe96ec382dacd249d19fdb3e2020eb3e02f4f3d9c7b8779d8e15
95753361ed6f492242a0eba172e4a30cfd62d0e335044cb3e7f722c9e699509f9cdd59a
ed90fe1b8bd40799f88f8555e21edb6f46269868481d
I/System.out(29638): public exponent: 10001

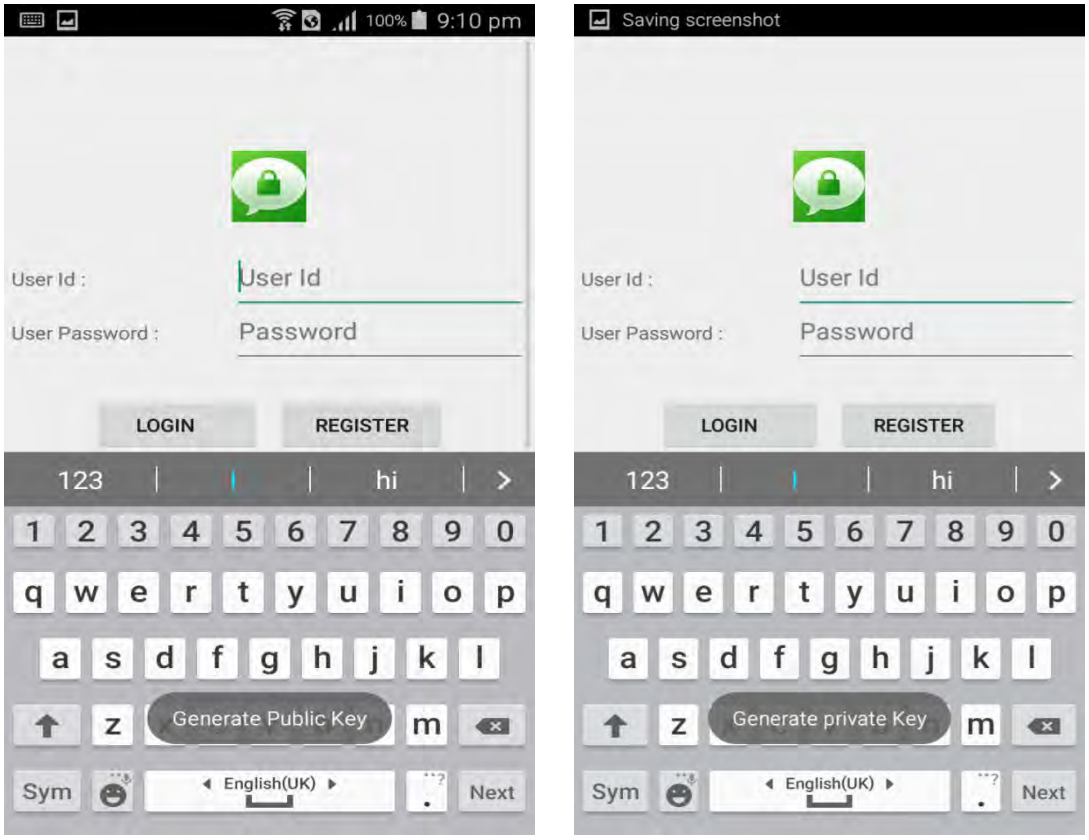

Figure 5.5 Screen shots of key generation during Registration

I/System.out(29638): dir /storage/emulated/0/Download
I/System.out(29638): privateKey: RSA Private CRT Key
I/System.out(29638):modulus:
f8270327ed0fc76eeae945c64c8372068459624613256b44e0142ac2c4718d16ab8684
95e033d137a3726fe3abe96ec382dacd249d19fdb3e2020eb3e02f4f3d9c7b8779d8e15
95753361ed6f492242a0eba172e4a30cfd62d0e335044cb3e7f722c9e699509f9cdd59a
ed90fe1b8bd40799f88f8555e21edb6f46269868481d
I/System.out(29638): public exponent: 10001
I/System.out(29638): private exponent:
88d5aa4dd7d0bd9e11f6f3bcb6f03c6e0f88e241d5c8ea6f9e4840940992bbc3e66d54a
af8685a539dd33a9386763fd79b4e5f9472f4a40b0ef3277c548081e65768edd95ad22f
19d4a9f6fbf2fec215cecdd2836aba7b533bc0cadccf8763e4c41c5bb64b0e263b220484
55571044e57ef149055abe129bc3d196e38cc7a1c1

I/System.out(29638): primeP:
ff026df761b33f66475d5ef4af589010ea29309e0ef558f8e6549ff3f986d0abd9e14cb4f
6ddb24e8b8c4eb712d24eef4ef76cc8db0fbb9fd078808f8548b499
I/System.out(29638): primeQ:
f91dc3adac454093cf14fad65016318e6ec418198d59a7df8916541e26996d28d7db905
d97d636dae2112eb2f62b78ab6624d1f937c9851517979637ff065e25
I/System.out(29638): primeExponentP:
c2b5a0315b2ffa20a62cecebf756afe9ba6168f5c6861f412cdd40a490ca175aa02a1edfa
dfee8be805ec95b8fcb74e9d469a76e559555b5590029696c636f41
I/System.out(29638): primeExponentQ:
61580123b74264669dafdc08acc8cf5a91ebec62e7ba16002268683c53e6f621d49f867
d1c1ce7d8862822b069c338e6f906004abb42f2cef82612dc14ba5f45
I/System.out(29638): crtCoefficient:
6ed2ee4142d716c76b0c7a4cf47ecaae73653456c9271e8ab8f4c4a6b917d3b453d0f6c
3d2c55e138633f030e9fa338b7fac87c4aa3369c062e25a73f56326de

Figure 5.6 Screen shots of Registration Success Message

2. The Public key along with user info sends to server and save in a file as String as:
   Labony~lab~123~MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBg
   QD4JwMn7Q/HburpRcZMg3IGhFliRhMla0TgFCrCxHGNFquGhJXgM9E3

o3Jv46vpbsOC2s0knRn9s+ICDrPgL089nHuHedjhWVdTNh7W9JIkKg66Fy
5KMM/WLQ4zUETLPn9yLJ5plQn5zdWa7ZD+G4vUB5n4j4VV4h7bb0Ym
mGhIHQIDAQAB

3. Private Key and Public Key are saved in user mobile phone set in two different file named PrivateKey.txt and PublicKey.txt as string.

4. Finally user get the Registration Successful message from the server

5.4.2 User Login

Once user successfully done the registration process, user can directly Login to the application. Server will verify the user information and send Login Successful message and the user list to the client app and it will open a new activity with user list.

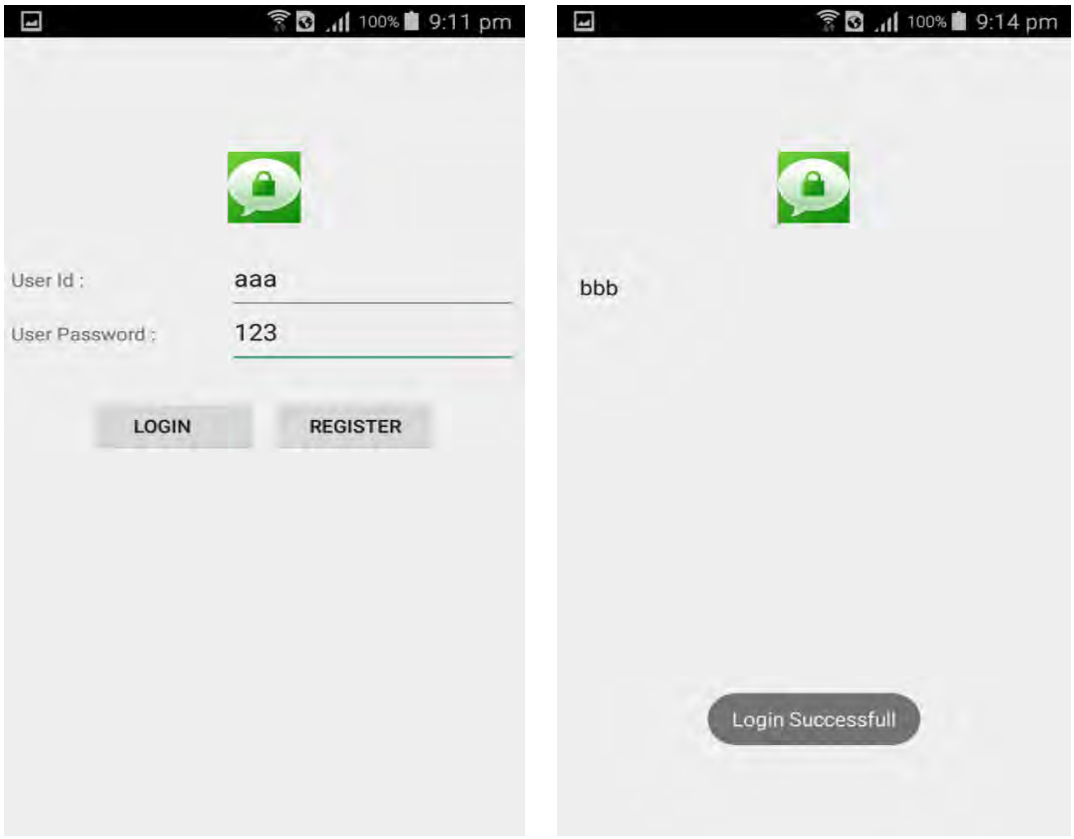

Figure 5.7 Screen shots of Login process

5.4.3 Message sending process

1. Logged user firstly select the receiver to whom user want to send message. Sender type the text and click send button.
2. Before sending message, client app will send a key Request to the server as:

   I/System.out(22833): userMsg infomation: hi

   D/HybiParser(22833): Creating frame for:

   {"flag":"keyreq","sessionId":"2","receiverId":"aaa","userId":"bbb"}

Sender ‘bbb’ Receiver ‘aaa’

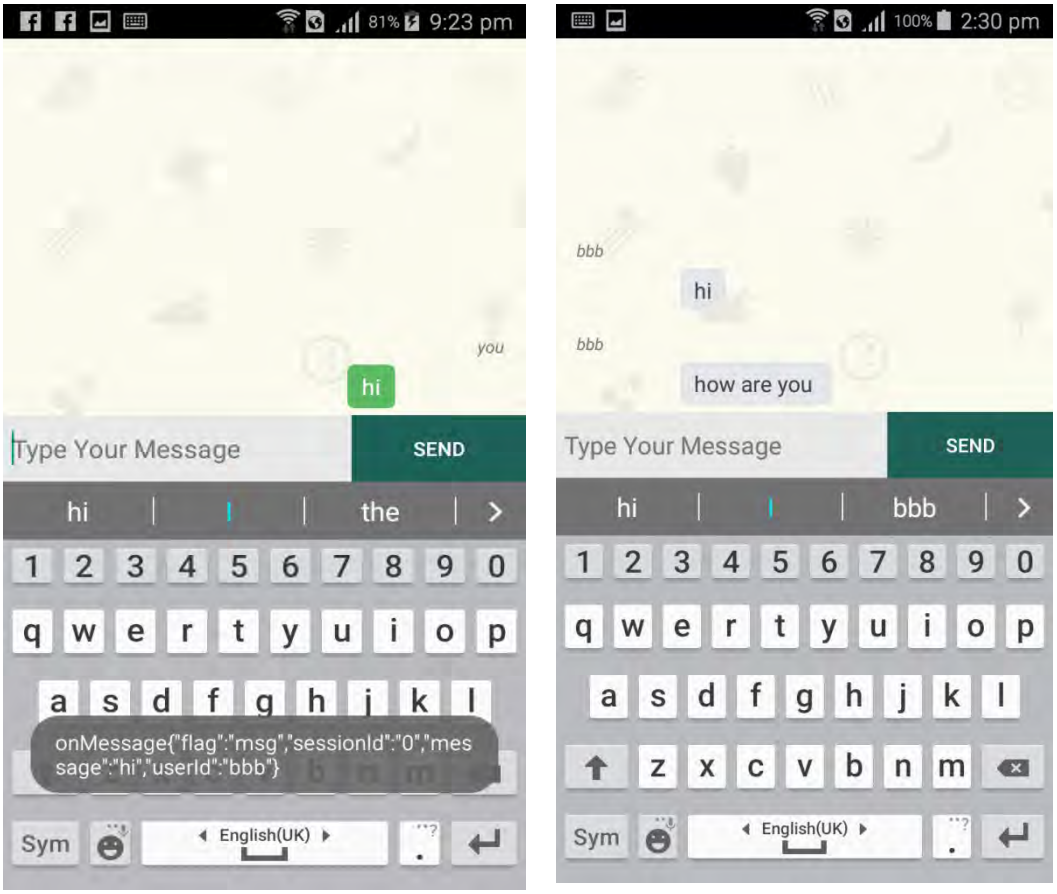

Figure 5.8 Screen shots of message send and receive

3. Get public key from server as String

   I/System.out(22833): keyFromServer:

   MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDXL+7nv

   Sr4j/PK/7Stn0HzBjfrPFFxdbtywrMt+jCCRB2L8r3N+/Ffojbb2owyg

   whWbIG7fRkEOdHrrWoiufAx1jc+TRp5MIaxmyv0jeKx3H19Y3H6/

wVCDMlpEVUnMYLhl/fXPWNbW2OzVehFIcoPuaBr+jLhutLTbF0
sZzfKTQIDAQAB

4. Retrieve Public key from the String

   I/System.out(22833): In loadPublicKey

   I/System.out(22833): publicKey RSA Public Key

   I/System.out(22833): modulus:
   d72feee7bd2af88ff3caffb4ad9f41f30637eb3c517175bb72c2b32dfa3082441d
   8bf2bdcdfbf15fa236dbda8c328308566c81bb7d190439d1ebad6a22b9f031d63
   73e4d1a793086b19b2bf48de2b1dc7d7d6371faff05420cc9691155273182e19
   7f7d73d635b5b63b355e84521ca0fb9a06bfa32e1bad2d36c5d2c6737ca4d

   I/System.out(22833): public exponent: 10001

5. Encrypt the user message with the public key and will send to the server

   I/System.out(22833): In eccryptedMsg:
   sj5TavjJ5ds9tIBemVq8fBeCPFqAoZsz4IGoo1V+Z6pUqnkKoAF60jOuDFjx
   GpOCn+Kyo+KfAJa1

   D/HybiParser(22833): Creating frame for:
   {"flag":"Msg","sessionId":"2","userId":"bbb","message":"sj5TavjJ5ds9tIBem
   Vq8fBeCPFqAoZsz4IGoo1V+Z6pUqnkKoAF60jOuDFjxGpOCn+Kyo+KfA
   Ja1\nrp\/dEOAu3Nmk3SpFPGiR6XovLdzI920\/BdXg8+oSozE5Jh91TEeKa
   PCoAXud1+wzI1vUyKQrrGEa\n7gjRlBPjULWyPQNayOI=\n","receiverId"
   :"aaa"}

6. Receiver receive the encrypted message and will decrypt it with private key and will get the original sender message

   I/System.out(22833): Encrypt str msg coming from server
   sj5TavjJ5ds9tIBemVq8fBeCPFqAoZsz4IGoo1V+Z6pUqnkKoAF60jOuDFjx
   GpOCn+Kyo+KfAJa1

   I/System.out(22833): Decrypt the messege

   I/System.out(22833):
   rp/dEOAu3Nmk3SpFPGiR6XovLdzI920/BdXg8+oSozE5Jh91TEeKaPCoA
   Xud1+wzI1vUyKQrrGEa

   I/System.out(22833): userMsg: hi

5.5 Analysis

The basic feature of popular Instant Messaging application is that for security they all are fully dependent on server. These applications rely on servers to store user related information, key generation, key management process, and also the encryption of confidential information. This is a major weakness of Instant Messaging system. Because there always has a chance to malicious attacks in the server side. Our proposed system is the best possible solution for the above mentioned drawback. In our system, we generate the public private key pair in android client and also do the encryption decryption process which restricts the server to have access or store the private keys and server also does not know what messages users are sending. This feature makes our system more safe and sound. Moreover, there is no chance to know the private keys of the client; it is harder to crack the encrypted messages than traditional applications. It is also significant that, if anyone tries to make the communication susceptible by login from another device, cannot decrypt the messages. Because the private key for that user is stored to the user's device.

Figure 5.9 Screen Shot of Unique Private Keys

After successful implementation of the system, we have verified our key generation system as keys will be unique each time it will be generated. We generate the key pair thousand times with two different users ID from a single client and store it in a excel file. Again do the same process with another two different user ID from another client device. Finally marge the excel files and get total four thousand keys. Then using the excel remove duplicate feature, we get the result as like figure 5.9 which shows no duplicate values found.

5.6 Summary

In our system, we generate the public private key pair in android client. The encryption decryption process is done in the client-side which restricts the server access to the private keys. Since the server does not have access to the private keys of the clients, it can only view encrypted messages. Thus, secure communication is ensured. To verify for possible key duplication, we generated the key pair four thousand times and stored it in a excel file. No duplicate key values are found. Although we generate the keys in client side, no chance to get same key pair by regenerating them.

CHAPTER 6

Conclusion and Future work

6.1 Conclusion

Mobile phones have emerged as an import technology in our everyday life and people are heavily depending on these small devices in their daily routines. Talking about their usages in daily communication through mobiles is quite common. To have more reliable communication through the untrusted servers, there is a need to have much security for such applications. For such a communication, depending upon server is not as efficient. Considering this assumption, there is a need of some security mechanisms for such mobile applications, which can provide security for user message. This can only be achieved by hiding secrete information from the server.

In this project, we have successfully developed an application which generates the keys at client side. We have developed this feature in a messaging application. This feature could easily be integrated with other applications as like financial mobile application and banking/payment applications. Our application allows a user to securely generate cryptographic key pair which is used to encrypt and decrypt user data, such as text messages, PIN/Password and transaction info. These keys would be only generated at registration time. Only public key is available at server for all users.

In short, our developed system has formed the basis for security of android messaging applications and proper usage of our concept could increase the security of keys as well as information and thus the users does not need to trust to the servers.

6.2 Future work

In this project, we use very basic information for user registration process and user private key is stored in user's mobile phone memory. To make this application more reliable, we will improve our registration information like date of birth, nationality,

cell phone number etc. As the most important part of our project is to protect user private key, we will put more concern on it. Now, Keys are stored in mobile SD card. SD card is also at risk of being compromised. To ensure the security of the stored private key, we can develop the feature of eSE (embedded Secure Element). By which, we ensure the security requirements of key management system.

The eSE is a tamper-proof chip available in different sizes and designs, embedded in any mobile device. It ensures the data is stored in a safe place and information is given to only authorize applications and people. It is like a personal ID for the end-user and for the device itself.

References

[1] Okafor, O., “Is it really secure? Evaluating the Security of Open Source Instant Messaging (IM) Applications”, http://www.academia.edu/10489985/Is_it_really_secure_Evaluating_the_Security_of_Open_Source_Instant_Messaging_IM_Applications, 2016. [Accessed 05 March 2017].

[2] Jennings, R.B., Nahum, E.M., Olshefski, D.P., Saha, D., Shae, Z.Y. and Waters, C., "A study of internet instant messaging and chat protocols", *IEEE Network*, *20*(4), pp.16-21

[3] Agar, J., "*Constant touch: A global history of the mobile phone*". Icon Books Ltd, London, 2013

[4] "The Evolution of Cell Phones", https://funalive.com/articles/the-evolution-of-cell-phones_W3M.html, [Last access on 02 Mar. 2017].

[5] "Phones Review", http://www.phonesreview.co.uk/2016/06/08/samsung-galaxy-s8-release-date-could-bring-a-4k-smartphone-display/, [Last access on 02 March 2017].

[6] Android, https://www.android.com/history/#/lollipop. [Accessed 15 March 2017].

[7] Nimodia, C. and Deshmukh, H.R., "Android operating system", Software Engineering, 3(1), pp.10-13, 2012.

[8] Apple Inc., "Apple Info", http://www.apple.com/about/, [Accessed 01 March 2017].

[9] Microsoft, https://www.microsoft.com/nl-NL/store/apps/windows-phone?rtc=1,2017. [Accessed 01 March 2017].

[10] BlackBerry Limited., http://us.blackberry.com/software/smartphones/blackberry-10-os, 2017.

[Accessed 01 March 2017].

[11] Azam, J., "Symmetric Key Management for Mobile Financial Applications: A Key Hierarchy Approach", Independent thesis Advanced level (degree of Master (Two Years)), KTH, School of Information and Communication Technology (ICT), p.37, 2013.

[12] IDC, "Smartphone OS Market Share", http://www.idc.com/promo/smartphone-market-share/os,2016.[Accessed 20 February 2017].

[13] Naik, M., Sindkar, A., Benali, P., & Moralwar, C., "Secure and Reliable Data Transfer on Android Mobiles Using AES and ECC Algorithm", Degree Project, Dr. D.Y. Patil College of Engineering, Pune.

[14] Alliance, S.C., 2011. "The mobile payments and nfc landscape: A us perspective. Smart Card Alliance", pp.1-53.

[15] Jain, A.K. and Shanbhag, D., "Addressing Security and Privacy Risks in Mobile Applications", *IT Professional, 14*(5), pp.28-33.

[16] Kaushik, A., Kumar, A. and Barnela, M.,"Block Encryption Standard for Transfer of data", In *2010 International Conference on Networking and Information Technology*, pp. 381-385. IEEE, 2010.

[17] Forouzan, B.A. and Mukhopadhyay, D., 2011. *Cryptography and Network Security*. McGraw-Hill Education.

[18] Stallings, W., 2006. Cryptography and network security: principles and practices. Pearson Education India.

[19] Mannan, M. and van Oorschot, P.C., "Secure public instant messaging: A survey", *Proceedings of Privacy, Security and Trust*, 2004.

[20] Schneier, B., 2000. *Applied Cryptography Second Edition: protocols,*

algorithms, and source. *Beijing: China MachinePress*.

[21] Chandramouli, R., Iorga, M. and Chokhani, S., "Cryptographic key management issues and challenges in cloud services", In *Secure Cloud Computing*, pp. 1-30, Springer New York, 2014.

[22] Lehtonen, S. and Parssinen, J., "Pattern Language for Cryptographic Key Management", In EuroPLoP, pp. 245-258. 2002.

[23] Jacobson, G., "The Public Key Muddle–How to Manage Transparent End-to-end Encryption in Organizations", In *ISSE 2015*, pp. 25-35. Springer Fachmedien Wiesbaden, 2015.

[24] Elbaz, L., "Using public key cryptography in mobile phones", *Discretix Technologies Ltd., White Paper* (2002).

[25] Neidhardt, E., "Asymmetric Cryptography for Mobile Devices", *Service-centric Networking*, pp.1-12, 2011.

[26] Preetha, M. and Nithya, M., "A study and performance Analysis of RSA algorithm", *IJCSMC Research Article* (2013).

[27] Arnold, K., Gosling, J., Holmes, D. and Holmes, D., 2000. The Java programming language (Vol. 2)., California 95054 U.S.A.

[28] The Eclipse Foundation, "Eclipse Kepler SR2 Packages", http://www.eclipse.org/downloads/packages/release/Kepler/SR2, 2017. [Accessed 15 March 2017].

[29] TechTarget, "Step-by-step guide to Android development with Eclipse", http://www.theserverside.com/tutorial/Step-by-step-guide-to-Android-development-with-Eclipse, 2000-2017. [Accessed 15 March 2017].

[30] Demo Source and Support, "Java JSON Tutorial - JSON Introduction", http://www.java2s.com/Tutorials/Java/JSON/index.htm. [Accessed 15 March 2017].

Glossary

SocketServer.java

```
package my.chat.info;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.URLDecoder;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;
import javax.websocket.OnClose;
import javax.websocket.OnMessage;
import javax.websocket.OnOpen;
import javax.websocket.Session;
import javax.websocket.server.ServerEndpoint;
import org.json.JSONException;
import org.json.JSONObject;
import com.google.common.collect.Maps;
import com.google.gson.Gson;
@ServerEndpoint("/chat")
public class SocketServer {
   // set to store all the live sessions
   private static final Set<Session> sessions = Collections.synchronizedSet(new
                                                                HashSet<Session>());
   // Mapping between session and person name
   private static final HashMap<String, String> nameSessionPair = new
                                                        HashMap<String, String>();
   private JSONUtils jsonUtils = new JSONUtils();
   String  name = "";
   String  userName = null,
                    userId= null,
                    receiverId= null,
                    password= null,
                    userPass= null,
                    tag= null,
                    result=null,
                    publicKey=null,
                    keySend=null,
                    userMsg = null;
   // Getting query params
   public static Map<String, String> getQueryMap(String query) {
      System.out.println("In getQueryMap  SocketServer class  ");
      Map<String, String> map = Maps.newHashMap();
      if (query != null) {
```

```
        String[] params = query.split("&");
        for (String param : params) {
           String[] nameval = param.split("=");
           map.put(nameval[0], nameval[1]);
        }
    }
    System.out.println("In getQueryMap return Map  " + map);
    return map;
}
/**
 * Called when a socket connection opened
 * @throws IOException
 * */
@OnOpen
public void onOpen(Session session) throws IOException {
    System.out.println(session.getId() + " has opened a connection");
    Map<String, String> queryParams = getQueryMap(session.getQueryString());
    if (queryParams.containsKey("name")) {
   // Getting client name via query param
        name = queryParams.get("name");
        System.out.println("Client Information in Server:  "+name);
        try {
        name = URLDecoder.decode(name, "UTF-8");
        } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
        }
        //get the operation tag
        if (name != null) {
        String[] params = name.split("~");
        tag=params[0];
        }
        //For new user Login
        if(tag.equalsIgnoreCase("Log"))
        {
        if (name != null) {
        String[] params = name.split("~");
        userId=params[1];
        password=params[2];
        }
        userPass = userId+"~"+password;
        result=UserInfoReadWrite.searchUsingBufferedReader(userPass);
        System.out.println("result:  "+result);
        if(result.equalsIgnoreCase("true"))
        {
        System.out.println("In Login true");
        List<String>  userList=UserInfoReadWrite.readUserlist(userId);
        String userlst= new Gson().toJson(userList);
```

```
  System.out.println("In Login true userList:"+userList);
  // Mapping client name and session id
  nameSessionPair.put(session.getId(),userId) ;
  // Adding session to session list
  sessions.add(session);
  try {
         // Sending session id to the client that just connected
          session.getBasicRemote().sendText(
          jsonUtils.getClientDetailsJson(session.getId(),"Login Successfull",userlst));
          System.out.println("In try send client details");
 } catch (IOException e) {
          e.printStackTrace();
 }
  // Notifying all the clients about new person joined
  sendMessageToClient(session.getId(), userId,  "joined conversation!",tag);
  System.out.println("sessions in SocketServer:"+nameSessionPair);
}
    else
         {
          System.out.println("onOpen: else Before sendMessageToClient");
          sendMessageToClient(session.getId(), userId, "Login Failed",  tag);
          System.out.println("onOpen: else After sendMessageToClient");
         }
}
//For new User registration
else if(tag.equalsIgnoreCase("Reg"))
{
         if (name != null) {
          String[] params = name.split("~");
          userName=params[1];
          password=params[2];
          userId=params[3];
          publicKey=params[4].replace(" ", "+");
         }
System.out.println("onMessage: publicKey:"+publicKey);
userPass = userName+"~"+userId+"~"+password+"~"+publicKey;
UserInfoReadWrite.writeInfoToFile(userPass);
System.out.println("onMessage: Before sendMessageToClient");
// Mapping client name and session id
nameSessionPair.put(session.getId(),userId) ;
// Adding session to session list
sessions.add(session);
sendMessageToClient(session.getId(), userName, "Registration Successfull",  tag);
sessions.remove(session.getId());
System.out.println("onMessage: After sendMessageToClient");
}} } }
```

JSONUtils.java

```
package my.chat.info;
import org.json.JSONException;
import org.json.JSONObject;
public class JSONUtils {
// flags to identify the kind of json response on client side
private static final String FLAG_LOG = "log" ,FLAG_SELF = "self",
FLAG_REG = "reg",FLAG_KEYREQ = "keyreq",
FLAG_MESSAGE = "msg", FLAG_EXIT = "exit";
public JSONUtils() {
}
/**
 * Json when client needs it's own session details with user list
 * */
public String getClientDetailsJson(String sessionId, String message, String userlst) {
String json = null;
        try {
                JSONObject jObj = new JSONObject();
                jObj.put("flag", FLAG_SELF);
                jObj.put("sessionId", sessionId);
                jObj.put("message", message);
                jObj.put("userlst", userlst);
                json = jObj.toString();
        } catch (JSONException e) {
                e.printStackTrace();
        }       return json;
}
/**
 * Json to notify the clients about Login
 * */
public String getNewClientLoginJson(String sessionId, String userId, String message
,int onlineCount)
{
        String json = null;
        System.out.println("In getNewClientLoginJson in   JSONUtils"+sessionId);
        try {
                JSONObject jObj = new JSONObject();
                jObj.put("flag", FLAG_LOG);
                jObj.put("userId", userId);
                jObj.put("sessionId", sessionId);
                jObj.put("message", message);
                jObj.put("onlineCount", onlineCount);
                json = jObj.toString();
        } catch (JSONException e) {
                e.printStackTrace();
        }
        System.out.println("In getNewClientLoginJson in   JSONUtils json   " +json);
```

```
        return json;
}
/**
 * Json to request for receiver Public key from Server
 * */
public String getPubKeyJson(String sessionId, String userId,String message,String
receiverId)
{
        String json = null;
        System.out.println("In getPubKeyJson in   JSONUtils");
        try {
                JSONObject jObj = new JSONObject();
                jObj.put("flag", FLAG_KEYREQ);
                jObj.put("userId", userId);
                jObj.put("sessionId", sessionId);
                jObj.put("message", message);
                jObj.put("receiverId", receiverId);
                json = jObj.toString();
        } catch (JSONException e) {
                e.printStackTrace();
        }
        System.out.println("In getPubKeyJson in   JSONUtils json   " +json);
        return json;
}
/**
 * JSON when message needs to be sent to all the clients
 * */
public String getSendAllMessageJson(String sessionId, String userId, String
message)
{
        String json = null;
        System.out.println("In getSendAllMessageJson in   JSONUtils");
        try {
                JSONObject jObj = new JSONObject();
                jObj.put("flag", FLAG_MESSAGE);
                jObj.put("sessionId", sessionId);
                jObj.put("userId", userId);
                jObj.put("message", message);
                json = jObj.toString();
        } catch (JSONException e) {
                e.printStackTrace();
        }
       System.out.println("In getSendAllMessageJson in   JSONUtils json  " +json );
        return json;
}
/**
 * Json to notify the clients about registration
```

```
 * */
public String getNewClientRegJson(String sessionId, String userName, String
message)
{
	String json = null;
	System.out.println("In getNewClientRegJson in   JSONUtils");
	try {
		JSONObject jObj = new JSONObject();
		jObj.put("flag", FLAG_REG);
		jObj.put("name", userName);
		jObj.put("sessionId", sessionId);
		jObj.put("message", message);

		json = jObj.toString();
	} catch (JSONException e) {
		e.printStackTrace();
	}
	System.out.println("In getNewClientRegJson in   JSONUtils json   " +json);
	return json;
}
/**
 * Json when the client exits the socket connection
 * */
public String getClientExitJson(String sessionId, String name,
		String message, int onlineCount) {
		String json = null;
	try {
		JSONObject jObj = new JSONObject();
		jObj.put("flag", FLAG_EXIT);
		jObj.put("name", name);
		jObj.put("sessionId", sessionId);
		jObj.put("message", message);
		jObj.put("onlineCount", onlineCount);
		json = jObj.toString();
	} catch (JSONException e) {
		e.printStackTrace();
	}	return json;
}}
```

UserInfoReadWrite.java

```
package my.chat.info;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
```

```
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;
import org.apache.commons.lang3.StringUtils;
public class UserInfoReadWrite{
public static void writeInfoToFile(String userInfo){
String path=UserInfoReadWrite.class.getResource("").getPath();
File file = new File(path, "UserInfo.txt");
try {
        FileOutputStream f = new FileOutputStream(file, true);
        PrintWriter pw = new PrintWriter(f);
        pw.println("\n"+ userInfo);
        pw.close();
        f.flush();
        f.close();
} catch (FileNotFoundException e) {
        e.printStackTrace();
} catch (IOException e) {
        e.printStackTrace();
}  }
public static String searchUsingBufferedReader(String searchQuery) throws
IOException
{
String path=UserInfoReadWrite.class.getResource("UserInfo.txt").getPath();
searchQuery = searchQuery.trim();
BufferedReader br = null;boolean result=false;
try
{
        br = new BufferedReader(new InputStreamReader(new
                                                FileInputStream(path)));
        String line;
        while ((line = br.readLine()) != null)
        {
                if (line.contains(searchQuery))
                {
                        result=true;
                        break;
                }
                else
                {               result=false;           }
        }
}catch (Exception e)
{
        System.out.println("Exception while closing bufferedreader " + e.toString());
```

```
}
finally
{
	try
	{
		if (br != null)
			br.close();
		if(result)
			return "true";
		else return "Fail";
	}
	catch (Exception e)
	{
	System.err.println("Exception while closing bufferedreader " + e.toString());
	}}
return null;
}
public static String searchUserPublicKey(String userId) throws IOException
{
	String path=UserInfoReadWrite.class.getResource("UserInfo.txt").getPath();
	System.out.println("in searchUserPublicKey path: "+path);
	String line = null;
	userId = userId.trim();
	BufferedReader br = null;String result=null;
	try
	{
	br = new BufferedReader(new InputStreamReader(new
	FileInputStream(path)));
	while ((line = br.readLine()) != null)
	{
	if (line.contains(userId))
	{
		result=line;
		break;
	}
	else
	{
	System.out.println("No User key found in searchUserPublicKey " );
	}	}
}catch (Exception e)
{System.out.println("Exception while closing searchUserPublicKey " + e.toString());
}
finally
{
try
{
	if (br != null)
```

```
        br.close();
        if(result != null)
        return result;
        else return "Fail";
        }
catch (Exception e)
{
System.err.println("Exception while closing searchUserPublicKey " + e.toString());
}
}
return line;
}
public static List<String> readUserlist(String loguser) throws IOException
{
String filename=UserInfoReadWrite.class.getResource("UserInfo.txt").getPath();
String regUsersinServer;
FileReader fileReader = new FileReader(filename);
BufferedReader bufferedReader = new BufferedReader(fileReader);
ArrayList<String> listOfUsers = new ArrayList<String>();
String line = "";
        while ((line = bufferedReader.readLine()) != null)
        {
        if(StringUtils.isBlank(line.trim()))     {}
        else
        {
                String[] parts = line.split("~");
                regUsersinServer = parts[1];
                if(regUsersinServer.equals(loguser))
                {}
                else
                listOfUsers.add(regUsersinServer.trim());
                }
        }
bufferedReader.close();
}}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.info.mychatapp"
   android:versionCode="1"
   android:versionName="1.0" >
   <uses-sdk
      android:minSdkVersion="13"
      android:targetSdkVersion="22" />
   <uses-permission android:name="android.permission.INTERNET" />
```

```
        <uses-permission
        android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".LoginActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".MessageActivity"
            android:label="@string/title_activity_message" >
        </activity>
        <activity
            android:name=".RegActivity"
            android:label="@string/title_activity_reg" >
        </activity>
        <activity
            android:name=".ActiveUserListActivity"
            android:label="@string/title_activity_active_user_list" >
        </activity>
    </application>
</manifest>
```

activity_reg.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:custom="http://schemas.android.com/apk/res-auto"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:fillViewport="true"
    tools:context=".AutoScrollTextViewActivity" >
    <RelativeLayout android:layout_width="fill_parent"
        android:layout_height="wrap_content" >
        <LinearLayout android:id="@+id/linerLayoutMenu"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:gravity="center"
                    android:layout_gravity="center"
                    android:paddingTop="88dp"
```

```
            android:paddingBottom="25dp" >
    <ImageButton android:id="@+id/imgBtnLOGO"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:background="@android:color/transparent"
            android:src="@drawable/ic_launcher"
             android:gravity="center"
            android:layout_gravity="center"
            android:contentDescription="Logo" />
</LinearLayout>
<LinearLayout android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:layout_below="@id/linerLayoutMenu" >
    <TableRow android:id="@+id/tblRowUName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center" >
            <TextView android:id="@+id/txtVwUserName"
                    android:text="User Name :       "
                    android:layout_width="150dp"
                    android:layout_height="wrap_content"
                    android:typeface="sans"
                     android:singleLine="true" />
            <EditText android:id="@+id/edtTxtUserName"
                    android:hint="User Name"
                    android:layout_width="200dp"
                    android:layout_height="wrap_content"
                    android:singleLine="true" />
    </TableRow>
            <TableRow android:id="@+id/tableRowUId"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content" >
            <TextView android:id="@+id/txtVwUId"
                    android:text="User Id : "
                    android:layout_width="150dp"
                    android:layout_height="wrap_content"
                    android:singleLine="true" />
            <EditText android:id="@+id/edtTxtUId"
                    android:hint="User Id"
                    android:layout_width="200dp"
                    android:layout_height="wrap_content"
                    android:singleLine="true" />
     </TableRow>
     <TableRow android:id="@+id/tableRowPassword"
            android:layout_width="wrap_content"
```

```
                android:layout_height="wrap_content" >
                <TextView android:id="@+id/txtVwPassword"
                        android:text="User Password : "
                        android:layout_width="150dp"
                        android:layout_height="wrap_content"
                        android:singleLine="true" />
                <EditText android:id="@+id/edtTxtPassword"
                        android:hint="Password"
                        android:layout_width="200dp"
                        android:layout_height="wrap_content"
                        android:singleLine="true" />
            </TableRow>
            <TableRow android:id="@+id/tableRow1"
                        android:layout_width="250dp"
                        android:maxWidth="280dp"
                        android:layout_height="wrap_content"
                        android:paddingTop="25dp"
                        android:gravity="center" >
                <Button android:id="@+id/btnExit"
                        android:layout_width="115dp"
                        android:layout_height="45dp"
                        android:ems="1"
                        android:paddingLeft="10dip"
                        android:paddingRight="26dip"
                        android:singleLine="true"
                        android:gravity="center"
                        android:text="Exit"
                        android:textAlignment="center"
                        tools:ignore="UnusedAttribute"  />
                    <ImageButton  android:layout_width="10dp"
                        android:layout_height="40dp"
                        android:src="@android:color/transparent"
                        android:background="@android:color/transparent"
                        android:contentDescription="" />
                    <Button android:id="@+id/btnReg"
                        android:layout_width="115dp"
                        android:layout_height="45dp"
                        android:ems="1"
                        android:paddingLeft="10dip"
                        android:paddingRight="26dip"
                        android:singleLine="true"
                        android:gravity="center"
                        android:text="Register"  />
            </TableRow>
        </LinearLayout>
    </RelativeLayout>
</ScrollView>
```

activity_login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   xmlns:custom="http://schemas.android.com/apk/res-auto"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:fillViewport="true"
   tools:context=".AutoScrollTextViewActivity" >
   <RelativeLayout android:layout_width="fill_parent"
      android:layout_height="wrap_content" >
      <LinearLayout android:id="@+id/linerLayoutMenu"
               android:layout_width="fill_parent"
               android:layout_height="wrap_content"
               android:gravity="center"
               android:layout_gravity="center"
               android:paddingTop="88dp"
               android:paddingBottom="25dp" >
         <ImageButton android:id="@+id/imgBtnLOGO"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:background="@android:color/transparent"
            android:src="@drawable/ic_launcher"
            android:gravity="center"
            android:layout_gravity="center"
            android:contentDescription="Logo" />
      </LinearLayout>
      <LinearLayout android:layout_height="fill_parent"
         android:layout_width="fill_parent"
         android:gravity="center_horizontal"
         android:orientation="vertical"
         android:layout_below="@id/linerLayoutMenu"
                  <TableRow android:id="@+id/tableRowUId"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content" >
                  <TextView android:id="@+id/txtVwUId"
                  android:text="User Id : "
                  android:layout_width="150dp"
                  android:layout_height="wrap_content"
                  android:singleLine="true" />
                  <EditText android:id="@+id/edtTxtUId"
                           android:hint="User Id"
                           android:layout_width="200dp"
                           android:layout_height="wrap_content"
                           android:singleLine="true" />
          </TableRow>
                  <TableRow android:id="@+id/tableRowPassword"
```

```
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" >
                <TextView android:id="@+id/txtVwPassword"
                        android:text="User Password : "
                        android:layout_width="150dp"
                        android:layout_height="wrap_content"
                        android:singleLine="true" />
                <EditText android:id="@+id/edtTxtPassword"
                        android:hint="Password"
                        android:layout_width="200dp"
                        android:layout_height="wrap_content"
                        android:singleLine="true" />
            </TableRow>
                <TableRow android:id="@+id/tableRow1"
                android:layout_width="250dp"
                     android:maxWidth="280dp"
                     android:layout_height="wrap_content"
                      android:paddingTop="25dp"
                     android:gravity="center" >
                   <Button android:id="@+id/btnLogin"
                        android:layout_width="115dp"
                        android:layout_height="45dp"
                        android:ems="1"
                        android:paddingLeft="10dip"
                        android:paddingRight="26dip"
                        android:singleLine="true"
                        android:gravity="center"
                        android:text="Login"
                        android:textAlignment="center"
                        tools:ignore="UnusedAttribute"  />
                   <ImageButton  android:layout_width="10dp"
                         android:layout_height="40dp"
                         android:src="@android:color/transparent"
                        android:background="@android:color/transparent"
                        android:contentDescription="" />
                   <Button android:id="@+id/btnReg"
                     android:layout_width="115dp"
                              android:layout_height="45dp"
                              android:singleLine="true"
                              android:gravity="center"
                       android:text="Register"  />
                </TableRow>
        </LinearLayout>
          </RelativeLayout>
</ScrollView>
```

activity_active_user_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   xmlns:custom="http://schemas.android.com/apk/res-auto"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:fillViewport="true"
   tools:context=".AutoScrollTextViewActivity" >
   <RelativeLayout android:layout_width="fill_parent"
      android:layout_height="wrap_content" >
      <LinearLayout android:id="@+id/linerLayoutMenu"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:layout_gravity="center"
            android:paddingTop="88dp"
            android:paddingBottom="25dp" >
         <ImageButton android:id="@+id/imgBtnLOGO"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:background="@android:color/transparent"
            android:src="@drawable/ic_launcher"
            android:gravity="center"
            android:layout_gravity="center"
            android:contentDescription="Logo" />
      </LinearLayout>
      <LinearLayout android:layout_height="fill_parent"
         android:layout_width="fill_parent"
         android:gravity="center_horizontal"
         android:orientation="vertical"
         android:layout_below="@id/linerLayoutMenu"
               <TableRow android:id="@+id/tableRowUId"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content" >
                  <ListView android:id="@+id/users"
               android:layout_width="wrap_content"
               android:layout_height="match_parent"/>
          </TableRow>
      </LinearLayout>
         </RelativeLayout>
</ScrollView>
```

activity_message.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
```

```
    android:background="@drawable/tile_bg"
    android:orientation="vertical" >
    <ListView
        android:id="@+id/list_view_messages"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@null"
        android:divider="@null"
        android:transcriptMode="alwaysScroll"
        android:stackFromBottom="true">
    </ListView>
    <LinearLayout
        android:id="@+id/llMsgCompose"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@color/white"
        android:orientation="horizontal"
        android:weightSum="3" >
        <EditText
            android:id="@+id/edtTxtinputMsg"
            android:layout_width="0dp"
            android:layout_height="fill_parent"
            android:layout_weight="2"
            android:background="@color/bg_msg_input"
            android:textColor="@color/text_msg_input"
            android:paddingLeft="6dp"
            android:paddingRight="6dp"
            android:hint="Type Your Message"/>
        <Button
            android:id="@+id/btnSend"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="@color/bg_btn_join"
            android:textColor="@color/white"
            android:text="Send"
            android:layout_gravity="center"/>
    </LinearLayout>
</LinearLayout>
```

list_item_message_left.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
```

```
    android:paddingBottom="5dp"
    android:paddingTop="5dp"
    android:paddingLeft="10dp"
    android:paddingRight="10dp">
    <TextView
        android:id="@+id/lblMsgFrom"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="12sp"
        android:textColor="@color/lblFromName"
        android:textStyle="italic"
        android:padding="5dp"/>
    <TextView
        android:id="@+id/txtMsg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="16sp"
        android:layout_marginLeft="80dp"
        android:layout_marginRight="80dp"
        android:textColor="@color/title_gray"
        android:paddingLeft="10dp"
        android:paddingRight="10dp"
        android:paddingTop="5dp"
        android:paddingBottom="5dp"
        android:background="@drawable/bg_msg_from"/>
</LinearLayout>
```

list_item_message_right.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="end"
    android:orientation="vertical"
    android:paddingBottom="5dp"
    android:paddingRight="10dp"
    android:paddingLeft="10dp"
    android:paddingTop="5dp" >

    <TextView
        android:id="@+id/lblMsgFrom"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5dp"
        android:textColor="@color/lblFromName"
        android:textSize="12sp"
```

```
        android:textStyle="italic" />
    <TextView
        android:id="@+id/txtMsg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="80dp"
        android:layout_marginRight="80dp"
        android:background="@drawable/bg_msg_you"
        android:paddingBottom="5dp"
        android:paddingLeft="10dp"
        android:paddingRight="10dp"
        android:paddingTop="5dp"
        android:textColor="@color/white"
        android:textSize="16sp" />
</LinearLayout>
```

RegActivity.java

```
package com.info.mychatapp;
import java.io.IOException;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.util.List;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import com.codebutler.android_websockets.WebSocketClient;
import com.info.mychatapp.common.ActivityExt;
import com.info.mychatapp.common.Message;
import com.info.mychatapp.common.MessagesListAdapter;
import com.info.mychatapp.common.Session;
import com.info.mychatapp.common.Utils;
import com.info.mychatapp.keypairgenerator.KeyGenerator;
public class RegActivity extends ActivityExt implements OnClickListener{
        String userName = "";
        String userId = "";
        String password = "";
        EditText edtTxtUserName;
        EditText edtTxtUserId;
        EditText edtTxtPassword;
        Button btnExit;
        Button btnReg;
```

```
        TextView txtVwUserName;
        TextView txtVwUserId;
        TextView txtVwPassword;
        private String name = null;
        private String publicKey=null;
        private WebSocketClient client;
        // Chat messages list adapter
        private MessagesListAdapter adapter;
        private List<Message> listMessages;
        private ListView listViewMessages;
        private Utils utils;
        // JSON flags to identify the kind of JSON response
        private static final String TAG_SELF = "self",
                                    TAG_NEW = "new",
                                    TAG_MESSAGE = "message",
                                    TAG_EXIT = "exit";

@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_reg);
        Session.getInstance().setActivity(this);
        txtVwUserName = (TextView) findViewById(R.id.txtVwUserName);
        txtVwUserId = (TextView) findViewById(R.id.txtVwUId);
        txtVwPassword = (TextView) findViewById(R.id.txtVwPassword);
        edtTxtUserName = (EditText) findViewById(R.id.edtTxtUserName);
        edtTxtUserId = (EditText) findViewById(R.id.edtTxtUId);
        edtTxtPassword = (EditText) findViewById(R.id.edtTxtPassword);
        btnReg = (Button) findViewById(R.id.btnReg);
        btnReg.setOnClickListener(this);
        btnExit = (Button) findViewById(R.id.btnExit);
        btnExit.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View arg0) {
        System.out.println("In RegActivity for Exit  ");
        Intent intent = new Intent(RegActivity.this, LoginActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        intent.putExtra("Exit me", true);
        startActivity(intent);
        finish();
}
});
}
@Override
public void onClick(View arg0) {
userName = edtTxtUserName.getText().toString();
userId = edtTxtUserId.getText().toString();
```

```
password = edtTxtPassword.getText().toString();
if(userName != null && !userName.isEmpty()){
if (userId != null && !userId.isEmpty()) {
        if (password != null && !password.isEmpty()){
                //Key Generation and save in a file
                try {
                        /*for(int i=0;i<=99;i++)
                        {
                                publicKey=KeyGenerator.RSAKeyGen(this);
                        }*/
                        publicKey=KeyGenerator.RSAKeyGen(this);

                } catch (NoSuchAlgorithmException e) {
                        e.printStackTrace();
                } catch (NoSuchProviderException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }

        name="Reg~"+userName+"~"+password+"~"+userId+"~"+publicKey;
                System.out.println("name infomation:  "+name);

                utils = new Utils(getApplicationContext());
                ClientConnection.socketClient(name);
        }
        else
        Toast.makeText(getApplicationContext(), "Please Enter Password",
        Toast.LENGTH_SHORT).show();
}
else
        Toast.makeText(getApplicationContext(), "Please Enter User Id",
        Toast.LENGTH_SHORT).show();
}
else
        Toast.makeText(getApplicationContext(), "Please Enter User Name",
        Toast.LENGTH_SHORT).show();
}
@Override
public void doActivity() {
        System.out.println("In Reg Activity doActivity method:  ");
        String repMsgfromserver=Session.getInstance().getMessage();
        System.out.println("repMsgfromserver in doActivity:  "+repMsgfromserver);
        if(repMsgfromserver.equals("Registration Successfull"))
        {
        Intent LoginActivity=new Intent(RegActivity.this,LoginActivity.class);
        LoginActivity.putExtra("name", name);
```

```
        utils.storeSessionId(null);
        startActivity(LoginActivity);
}
else {}
}}
```

LoginActivity.java

```
package com.info.mychatapp;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import com.info.mychatapp.common.ActivityExt;
import com.info.mychatapp.common.Session;
import com.info.mychatapp.common.Utils;
public class LoginActivity extends ActivityExt implements OnClickListener{
        String userId = "";
        String password = "";
        private String userPass = null;
        private Utils utils;
        EditText edtTxtUserId;
        EditText edtTxtPassword;
        TextView txtVwUserId;
        TextView txtVwPassword;
        Button btnLogin;
        Button btnReg;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_login);
      Session.getInstance().setActivity(this);
      txtVwUserId = (TextView) findViewById(R.id.txtVwUId);
      txtVwPassword = (TextView) findViewById(R.id.txtVwPassword);
      edtTxtUserId = (EditText) findViewById(R.id.edtTxtUId);
      edtTxtPassword = (EditText) findViewById(R.id.edtTxtPassword);
      btnLogin = (Button) findViewById(R.id.btnLogin);
      btnLogin.setOnClickListener(this);
      btnReg = (Button) findViewById(R.id.btnReg);
      btnReg.setOnClickListener(this);
      btnReg.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
            Intent intent = new Intent(LoginActivity.this, RegActivity.class);
```

```
            startActivity(intent);
        }
    });
        if( getIntent().getBooleanExtra("Exit me", false)){
            finish();
            return; // add this to prevent from doing unnecessary stuffs
        }   }
@Override
public void onClick(View arg0) {
        userId = edtTxtUserId.getText().toString();
        password = edtTxtPassword.getText().toString();
        if (userId != null && !userId.isEmpty()) {
                if (password != null && !password.isEmpty()){
                        Session.getInstance().setUserId(userId);
                        String name = "Log~"+userId +"~"+password;
                        System.out.println("name infomation:  "+name);
                        utils = new Utils(getApplicationContext());
                        ClientConnection.socketClient(name);
                }
                else
                        Toast.makeText(getApplicationContext(), "Please Enter
                        Password", Toast.LENGTH_SHORT).show();
        }
        else
                Toast.makeText(getApplicationContext(), "Please Enter User Id",
                Toast.LENGTH_SHORT).show();
}
@Override
public void doActivity() {
        System.out.println("In Login Activity doActivity method:  ");
        String repMsgfromserver=Session.getInstance().getMessage();
        String activeUsersinServer = Session.getInstance().getUserList();
        System.out.println("repMsgfromserver:  "+repMsgfromserver);
        if (repMsgfromserver.equals("Login Failed"))
        {
System.out.println("In Login Activity doActivity method:  "+repMsgfromserver);
        }
else if(repMsgfromserver.equals("Login Successfull"))
{
        Intent ActiveUserListActivity=new
        Intent(LoginActivity.this,ActiveUserListActivity.class);
                        ActiveUserListActivity.putExtra("users",
                        activeUsersinServer);
                        startActivity(ActiveUserListActivity);
                }       }       }
```

ActiveUserListActivity.java

```
package com.info.mychatapp;
import java.util.ArrayList;
import java.util.List;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.Toast;
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;
import com.info.mychatapp.common.ActivityExt;
import com.info.mychatapp.common.Session;
import com.info.mychatapp.common.Utils;
public class ActiveUserListActivity extends ActivityExt {
	ListView usersList;
	ArrayList<String> listOfUser;
	private Utils utils;
@Override
protected void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	setContentView(R.layout.activity_active_user_list);
	Session.getInstance().setActivity(this);
	String users = getIntent().getStringExtra("users");
	listOfUser = new ArrayList<String>();
	java.lang.reflect.Type t = new TypeToken<List<String>>(){}.getType();
	listOfUser = new Gson().fromJson(users,t);
	usersList = (ListView) findViewById(R.id.users);
	usersList.setAdapter(new ArrayAdapter<String>(this,
	android.R.layout.simple_list_item_1,listOfUser));
	usersList.setOnItemClickListener(new OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
	Toast.makeText(getApplicationContext(), listOfUser.get(arg2),
	Toast.LENGTH_SHORT).show();
	String userToSendMsg=listOfUser.get(arg2);
	Intent MessageActivity=new
		Intent(ActiveUserListActivity.this,MessageActivity.class);
	MessageActivity.putExtra("userToSendMsg", userToSendMsg);
			startActivity(MessageActivity);
		}		});
}
@Override
```

```
public void doActivity() {
        System.out.println("In doActivity ActiveUserList");
        String repMsgfromserver=Session.getInstance().getMessage();
        System.out.println("In doActivity ActiveUserList
                                        repMsgfromserver:"+repMsgfromserver);
        if (repMsgfromserver.equals(" left conversation!"))
        {
         System.out.println("In ActiveUserList Activity doActivity method:
                                                        "+repMsgfromserver);

        Intent LoginActivity=new
                        Intent(ActiveUserListActivity.this,LoginActivity.class);
                        startActivity(LoginActivity);
        }               }}
```

MessageActivity.java

```
package com.info.mychatapp;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.List;
import javax.crypto.BadPaddingException;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import com.info.mychatapp.common.ActivityExt;
import com.info.mychatapp.common.Message;
import com.info.mychatapp.common.MessagesListAdapter;
import com.info.mychatapp.common.Session;
import com.info.mychatapp.common.Utils;
import com.info.mychatapp.keypairgenerator.EncryptDecryptMsg;
public class MessageActivity extends ActivityExt implements OnClickListener{
        private Utils utils;
        private ListView listViewMessages;
        private static MessagesListAdapter adapter;
        private static List<Message> listMessages;
        EditText edtTxtinputMsg;
        Button btnSend;
        String userMsg = "";
        String userId = "";
        String keyFromServer = "";
        String userToSendMsg;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_message);
        Session.getInstance().setActivity(this);
        listMessages = new ArrayList<Message>();
        adapter = new MessagesListAdapter(this, listMessages);
        edtTxtinputMsg = (EditText) findViewById(R.id.edtTxtinputMsg);
        listViewMessages = (ListView) findViewById(R.id.list_view_messages);
        listViewMessages.setAdapter(adapter);
        btnSend = (Button) findViewById(R.id.btnSend);
        btnSend.setOnClickListener(this);
        userToSendMsg = getIntent().getStringExtra("userToSendMsg");
        }
@Override
public void onClick(View arg0) {
        userMsg = edtTxtinputMsg.getText().toString();
        userId=Session.getInstance().getUserId();
        System.out.println("userToSendMsg In MessageActivity:
                                                        "+userToSendMsg);
        System.out.println("userMsg infomation:  "+userId+","+userMsg);
        utils = new Utils(getApplicationContext());
        //Get public key of receiver from server
        ClientConnection.sendMessageToServer(utils.getSendKeyReqJSON
                                                        (userId,userToSendMsg));
        //ClientConnection.sendMessageToServer(utils.getSendMessageJSON(userId
                                                                , userMsg));
        Session.getInstance().setMessage(userMsg);
        edtTxtinputMsg.setText("");
        }
/**
* Appending message to list view
* */
public static void appendMessage(final Message m) {
System.out.println("In appendMessage");
Session.getInstance().getActivity().runOnUiThread(new Runnable() {
@Override
public void run() {
        listMessages.add(m);
        adapter.notifyDataSetChanged();
        // Playing device's notification
        ClientConnection.playBeep();
                }                   });       }
@Override
public void doActivity() {
        String eccryptedMsg = null;
        System.out.println(" In doActivity:  ");
```

```
keyFromServer=Session.getInstance().getPubKeyfromServer();
System.out.println("keyFromServer:  "+keyFromServer);
//Encrypt message with receiver public key
try {
        System.out.println("In doActivity: in try  ");
        eccryptedMsg=EncryptDecryptMsg.RSAEncrypt(this);
        System.out.println("In  eccryptedMsg:  "+eccryptedMsg);

} catch (InvalidKeyException e) {
                e.printStackTrace();
} catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
} catch (NoSuchPaddingException e) {
                e.printStackTrace();
} catch (IllegalBlockSizeException e) {
                e.printStackTrace();
} catch (BadPaddingException e) {
                e.printStackTrace();
}
//Send Encrypted Message to server
ClientConnection.sendMessageToServer(utils.getSendMessageJSON(userId,
                                        eccryptedMsg,userToSendMsg));

}       }
```

ActivityExt.java

```
package com.info.mychatapp.common;
import android.app.Activity;
public abstract  class ActivityExt extends Activity{
        public abstract void doActivity();
}
```

Constant.java

```
package com.info.mychatapp.common;
public class Constant {
// JSON flags to identify the kind of JSON response
        public static final String  TAG_SELF = "self";
        public static final String      TAG_NEW = "new";
        public static final String      TAG_MESSAGE = "msg";
        public static final String      TAG_LOGOUT = "logout";
        public static final String      TAG_REG = "reg";
        public static final String      TAG_LOGIN = "log";
        public static final String      TAG_KEYREQ = "keyreq";
        public static final String      TAG_EXIT = "exit";
        public static final String      KEY_SESSION_ID = "sessionId";
        public static final String      KEY_SHARED_PREF = "MY_CHAT_APP";
}
```

Message.java

```
package com.info.mychatapp.common;
public class Message {
   private String fromName, message;
   private boolean isSelf;
public Message() {
}
public Message(String fromName, String message, boolean isSelf) {
   this.fromName = fromName;
   this.message = message;
   this.isSelf = isSelf;
}
public String getFromName() {
   return fromName;
}
public void setFromName(String fromName) {
   this.fromName = fromName;
}
public String getMessage() {
   return message;
}
public void setMessage(String message) {
   this.message = message;
}
public boolean isSelf() {
   return isSelf;
}
public void setSelf(boolean isSelf) {
   this.isSelf = isSelf;
}}
```

MessagesListAdapter.java

```
package com.info.mychatapp.common;
import java.util.List;
import com.info.mychatapp.R;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;
public class MessagesListAdapter extends BaseAdapter {
   private Context context;
   private List<Message> messagesItems;
   public MessagesListAdapter(Context context, List<Message> navDrawerItems) {
```

```
        this.context = context;
        this.messagesItems = navDrawerItems;
    }
    @Override
    public int getCount() {
        return messagesItems.size();
    }
    @Override
    public Object getItem(int position) {
        return messagesItems.get(position);
    }
    @Override
    public long getItemId(int position) {
        return position;
    }
    @SuppressLint("InflateParams")
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
/**
* The following list not implemented reusable list items as list items
* are showing incorrect data Add the solution if you have one
* */
   System.out.println("In MessagesListAdapter getView method");
   Message m = messagesItems.get(position);
    LayoutInflater mInflater = (LayoutInflater)
    context.getSystemService(Activity.LAYOUT_INFLATER_SERVICE);
        // Identifying the message owner
        if (messagesItems.get(position).isSelf()) {
            // message belongs to you, so load the right aligned layout
            convertView = mInflater.inflate(R.layout.list_item_message_right,  null);
        } else {
            // message belongs to other person, load the left aligned layout
            convertView = mInflater.inflate(R.layout.list_item_message_left, null);
        }
      TextView lblFrom = (TextView) convertView.findViewById(R.id.lblMsgFrom);
      TextView txtMsg = (TextView) convertView.findViewById(R.id.txtMsg);
      txtMsg.setText(m.getMessage());
      lblFrom.setText(m.getFromName());
      return convertView;
    }}
```

Session.java

```
package com.info.mychatapp.common;
import android.app.Activity;
public class Session {
          /**
           * Private Empty Constructor
```

```
	 */
	private Session() {}
	private  static Session _session=null;
	/**
	 * getInstance method returns a single instance of the class this.
	 * @return - It first checks if one instance is already created. If yes it returns
that instance else gets a new session and returns.
	 */
	public static Session getInstance()
	{
		if(_session==null)
		{	_session=new Session();	}
		return _session;
	}
	private String		sessionId;
	private String		userId;
	private String		userName;
	private String		userPassword;
	private String		serviceId;
	private String		message;
	private ActivityExt activity;
	Message m;
	private String  pubKeyfromServer;
	private String  userList;
	public String getUserList() {
		return userList;
	}
	public void setUserList(String userList) {
		this.userList = userList;
	}
	public String getSessionId() {
		return sessionId;
	}
	public void setSessionId(String sessionId) {
		this.sessionId = sessionId;
	}
	public String getUserId() {
		return userId;
	}
	public void setUserId(String userId) {
		this.userId = userId;
	}
	public String getUserName() {
		return userName;
	}
	public void setUserName(String userName) {
		this.userName = userName;
```

```
}
public String getUserPassword() {
        return userPassword;
}
public void setUserPassword(String userPassword) {
        this.userPassword = userPassword;
}
public String getServiceId() {
        return serviceId;
}
public void setServiceId(String serviceId) {
        this.serviceId = serviceId;
}
public String getMessage() {
        return message;
}
public void setMessage(String message) {
        this.message = message;
}
public ActivityExt getActivity() {
        return activity;
}
public void setActivity(ActivityExt activity) {
        this.activity = activity;
}
public Message getM() {
        return m;
}
public void setM(Message m) {
        this.m = m;
}
public String getPubKeyfromServer() {
        return pubKeyfromServer;
}
public void setPubKeyfromServer(String pubKeyfromServer) {
        this.pubKeyfromServer = pubKeyfromServer;
}}
```

Utils.java

```
package com.info.mychatapp.common;
import org.json.JSONException;
import org.json.JSONObject;
import android.content.Context;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
public class Utils {
   private Context context;
```

```
    private SharedPreferences sharedPref;
    private static final int KEY_MODE_PRIVATE = 0;
    public Utils(Context context) {
        this.context = context;
        sharedPref =
this.context.getSharedPreferences(Constant.KEY_SHARED_PREF,KEY_MODE_P
RIVATE);
    }
    public void storeSessionId(String sessionId) {
        Editor editor = sharedPref.edit();
        editor.putString(Constant.KEY_SESSION_ID, sessionId);
        editor.commit();
    }
    public String getSessionId() {
        return sharedPref.getString(Constant.KEY_SESSION_ID, null);
    }
  //JSON for User Login
    public String getSendLoginJSON(String userId , String password) {
        String json = null;
        try {
            JSONObject jObj = new JSONObject();
            jObj.put("flag", Constant.TAG_LOGIN);
            jObj.put("userId", userId);
            jObj.put("password", password);
            json = jObj.toString();
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return json;
    }
    //JSON for send Message
    public String getSendMessageJSON(String userId,String message,String
receiverId) {
        String json = null;
        try {
            JSONObject jObj = new JSONObject();
            jObj.put("flag", Constant.TAG_MESSAGE);
            jObj.put("sessionId", getSessionId());
            jObj.put("userId", userId);
            jObj.put("message", message);
            jObj.put("receiverId", receiverId);
            json = jObj.toString();
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return json;
    }
```

```
  //JSON for User Registration
    public String getSendRegJSON(String userName, String password, String userId,
String publicKey) {
        String json = null;
        try {
            JSONObject jObj = new JSONObject();
            jObj.put("flag", Constant.TAG_REG);
            jObj.put("sessionId", getSessionId());
            jObj.put("userName", userName);
            jObj.put("password", password);
            jObj.put("userId", userId);
            jObj.put("publicKey", publicKey);
            json = jObj.toString();
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return json;
    }
  //this method is for public key request for encryption
    public String getSendKeyReqJSON(String userId,String receiverId) {
        String json = null;
        try {
            JSONObject jObj = new JSONObject();
            jObj.put("flag", Constant.TAG_KEYREQ);
            jObj.put("sessionId", getSessionId());
            jObj.put("receiverId", receiverId);
            jObj.put("userId", userId);
            json = jObj.toString();
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return json;
    }
  //JSON for User Logout
    public String getSendLogoutJSON(String userId ) {
        String json = null;
        try {
            JSONObject jObj = new JSONObject();
            jObj.put("flag", Constant.TAG_LOGOUT);
            jObj.put("sessionId", getSessionId());
            jObj.put("userId", userId);
            json = jObj.toString();
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return json;
    }
```

```
}
```

WsConfig.java

```
package com.info.mychatapp.common;
public class WsConfig {
   public static final String URL_WEBSOCKET =
"ws://192.168.1.104:8080/MyChatServer/chat?name=";
}
```

KeyGenerator.java

```
package com.info.mychatapp.keypairgenerator;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.widget.Toast;
import java.io.IOException;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.SecureRandom;
import java.security.Security;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import com.apps.codec.binary.Base64;
public  class KeyGenerator {
        static KeyPairGenerator kpg;
   static KeyPair kp;
   static PublicKey publicKey;
   static PrivateKey privateKey;
   static String publicK,privateK;
        //generate Public Private key pair and return Public Key to MainActivity
        @SuppressLint("TrulyRandom") public static String RSAKeyGen (Activity
activity)
                        throws NoSuchAlgorithmException, IOException,
NoSuchProviderException{
                System.out.println("In RSAKeyGen() in KeyGenerator class");

                //Toast.makeText(activity.getApplicationContext(), "Start",
Toast.LENGTH_LONG).show();
                Security.addProvider(new
org.bouncycastle.jce.provider.BouncyCastleProvider());
        SecureRandom random = new SecureRandom();
        kpg = KeyPairGenerator.getInstance("RSA","BC");
      kpg.initialize(1024,random);
      kp = kpg.genKeyPair();
        //Generate Public Key and save in a file as text
```

```
            publicKey = kp.getPublic();
            X509EncodedKeySpec x509EncodedKeySpecPub = new
X509EncodedKeySpec(        publicKey.getEncoded());
                System.out.println("publicKey:   "+publicKey);
                publicK=new
String(Base64.encodeBase64(x509EncodedKeySpecPub.getEncoded()));
                //Save Public Key
                ReadWriteKeys.writePublicKeyToFile(publicK);

                Toast.makeText(activity.getApplicationContext(), "Generate Public
Key", Toast.LENGTH_LONG).show();
                //Generate Private Key and save in a file as text
            privateKey = kp.getPrivate();
            PKCS8EncodedKeySpec keySpec = new
PKCS8EncodedKeySpec(privateKey.getEncoded());
            System.out.println("privateKey:   "+privateKey);
            privateK=new String(Base64.encodeBase64(keySpec.getEncoded()));

            //Save Private Key
            ReadWriteKeys.writePrivateKeyToFile(privateK);
            Toast.makeText(activity.getApplicationContext(), "Generate private
Key", Toast.LENGTH_LONG).show();
        return publicK;
   }
}
```

ReadWriteKeys.java

```
package com.info.mychatapp.keypairgenerator;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import android.annotation.SuppressLint;
import android.content.Context;
public class ReadWriteKeys {
        /** Method to write ascii text characters to file on SD card. Note that you
must add a
          WRITE_EXTERNAL_STORAGE permission to the manifest file or this
method will throw
          a FileNotFound Exception because you won't have write permission. */
        public static void writePublicKeyToFile(String pubKToWrite){
```

```
        // Find the root of the external storage.
              System.out.println("line a");
        File root = android.os.Environment.getExternalStorageDirectory();
        File dir = new File (root.getAbsolutePath() + "/Download");
        System.out.println("dir    "+dir);
        File file = new File(dir, "PublicKeys.txt");
        try {
           FileOutputStream f = new FileOutputStream(file, true);
           PrintWriter pw = new PrintWriter(f);
                 pw.println("\n"+ pubKToWrite);
           pw.flush();
           pw.close();
           f.close();
        } catch (FileNotFoundException e) {
           e.printStackTrace();
        } catch (IOException e) {
           e.printStackTrace();
        }
     }
     public static void writePrivateKeyToFile(String priKToWrite){
        File root = android.os.Environment.getExternalStorageDirectory();
        File dir = new File (root.getAbsolutePath() + "/Download");
        System.out.println("dir    "+dir);
        //dir.mkdirs();
        File file = new File(dir, "PrivateKeys.txt");
        try {
           FileOutputStream f = new FileOutputStream(file,true);
           PrintWriter pw = new PrintWriter(f);
           pw.println("\n"+ priKToWrite);
           pw.flush();
           pw.close();
           f.close();
        } catch (FileNotFoundException e) {
           e.printStackTrace();
        } catch (IOException e) {
           e.printStackTrace();
        }
     }
     public static String readPrivateKeyFromFile() throws IOException
{
              System.out.println("In readPrivateKeyFromFile   ");
              File root = android.os.Environment.getExternalStorageDirectory();
              File dir = new File (root.getAbsolutePath() + "/Download");

              String path=dir+"/PrivateKeys.txt";
              String thisLine = null;
              File f = new File(path);
```

```
                if (f.exists())
                {
                FileReader fr = new FileReader(f);
                BufferedReader br = new BufferedReader(fr);

                        while ((thisLine = br.readLine()) != null)
                        {
                                System.out.println("In readPrivateKeyFromFile Private
key from client memory   :"+thisLine);
                                if(thisLine!=null && !thisLine.isEmpty() &&
!thisLine.equals(""))
                                break;
                        }
        }
        System.out.println("In readPrivateKeyFromFile Private key from client
memory   :"+thisLine);
        return thisLine;
   }
}
```

EncryptDecryptMsg.java

```
package com.info.mychatapp.keypairgenerator;
import java.io.IOException;
import java.security.GeneralSecurityException;
import java.security.InvalidKeyException;
import java.security.KeyFactory;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import com.apps.codec.binary.Base64;
import com.info.mychatapp.common.Session;
import android.annotation.SuppressLint;
import android.app.Activity;
public class EncryptDecryptMsg {
        static byte [] encryptedBytes, decryptedBytes,encryptedMsgBytes;
   static Cipher cipherEn,cipherDe ;
   static String encryptedMsg, decryptedMsg;
        static String strkey;
   static PublicKey publicKey;
   static PrivateKey privateKey;
        @SuppressLint("TrulyRandom")
```

```
        public static String RSAEncrypt (Activity activity) throws
NoSuchAlgorithmException, NoSuchPaddingException,
        InvalidKeyException, IllegalBlockSizeException, BadPaddingException
   {
        try {
                strkey=Session.getInstance().getPubKeyfromServer();
                        publicKey=loadPublicKey(strkey);
                        String plain=Session.getInstance().getMessage();
                        System.out.println("plain in RSAEncrypt:  "+plain);

                cipherEn = Cipher.getInstance("RSA");
                cipherEn.init(Cipher.ENCRYPT_MODE, publicKey);
              encryptedBytes = cipherEn.doFinal(plain.getBytes());
              System.out.println("encryptedBytes in RSAEncrypt:
"+encryptedBytes);
              encryptedMsg=BytetoStr.getenString(encryptedBytes);
        }
      catch (GeneralSecurityException e) {
                        e.printStackTrace();
                }
      return encryptedMsg;
   }
        public static PublicKey loadPublicKey(String stored) throws
GeneralSecurityException {
                //Convert PublicKeyString to Byte Stream
                byte[] sigBytes2 = Base64.decodeBase64(stored.getBytes());
                System.out.println(" In loadPublicKey fact.generatePublic(spec) after
byte conversion");
           // Convert the public key bytes into a PublicKey object
           X509EncodedKeySpec x509KeySpec = new
X509EncodedKeySpec(sigBytes2);
           KeyFactory keyFact = KeyFactory.getInstance("RSA", "BC");
           publicKey = keyFact.generatePublic(x509KeySpec);
           System.out.println(" In loadPublicKey publicKey before return
:"+publicKey);
           return publicKey;
        }
        public static String RSADecrypt(String enmessage)   throws
NoSuchAlgorithmException, NoSuchPaddingException,
        InvalidKeyException, IllegalBlockSizeException, BadPaddingException
   {
        //Retrieve Private key from file
        String strPrivateKey = null;
        try {
                strPrivateKey=ReadWriteKeys.readPrivateKeyFromFile();
```

```
                System.out.println("in RSADecrypt() after calling
ReadWriteKeys.readPrivateKeyFromFile()   : "+strPrivateKey);

        } catch (IOException e) {
                        e.printStackTrace();
                }
        try {
                        privateKey=loadPrivateKey(strPrivateKey);
                        encryptedMsgBytes=BytetoStr.getenByte(enmessage);
                cipherDe = Cipher.getInstance("RSA");
                        cipherDe.init(Cipher.DECRYPT_MODE, privateKey);
                        decryptedBytes = cipherDe.doFinal(encryptedMsgBytes);
                        decryptedMsg=new String(decryptedBytes);
                        System.out.println("Decrypted string in RSADecrypt() in
RSAKeyPairGen class....." + decryptedMsg);
        }catch (GeneralSecurityException e) {
                        e.printStackTrace();
                }return decryptedMsg;
   }
        public static PrivateKey loadPrivateKey(String strPrivateKey) throws
GeneralSecurityException {
                byte[] sigBytes = Base64.decodeBase64(strPrivateKey.getBytes());
      // extract the private key
      PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(sigBytes);
      KeyFactory kf = KeyFactory.getInstance("RSA","BC");
      privateKey = kf.generatePrivate(keySpec);
      System.out.println(privateKey);
          return privateKey;
        }       }
```

BytetoStr.java

```
package com.info.mychatapp.keypairgenerator;
import android.util.Base64;
public class BytetoStr {
        public static String getenString (byte [] enbyte)
        {
                String encryptedstr=Base64.encodeToString(enbyte,
Base64.DEFAULT);
                return encryptedstr;
        }
        public static byte[] getenByte (String encryptedstr)
        {
                byte [] decryptedbyte=Base64.decode(encryptedstr,
Base64.DEFAULT);
                return decryptedbyte;
        }
}
```