**CHAPTER THREE**

**DESIGN METHODOLOGY**

**3.1 System Overview**

The overall structure of the mobile android based secure intranet based, instant messenger, IM (a chatting/messaging system) platform is hereby presented. Encryption security (blowfish algorithm in this proposed IM system) is one of the key important feature of an IM platform. Combining encryption technique and the fact that the messages are transmitted over the intranet makes it uncommonly secure. What is being shared or discussed between the users must remain private, and no other person should be able to break into their chats on the platform. This is the reason why security feature of using intranet and other security techniques is given a serious undertone in the architecture of this IM chat platform. This chapter describes the design, architecture and implementation of secure chatting system and they will be succinctly explained. This architecture has to meet the requirements of confidentiality and integrity of messages, as well as privacy of users. The architecture includes a number of components and each component implemented as a separate entity.

**3.2 Architecture of the Secure Intranet Chat System**

**3.2.1 System Architecture**

This chat system whose architecture is of two major parts as shown in figures 3.1a and 3.1b, which are sender and receiver, via a local host web based server developed using PHP. However, these parts have subcomponents.



Users (E2EE)

between two verified

intranet message

User 1

User 2



**CA Server**



IM

Server



Figure 3.1a Overview of the Architecture

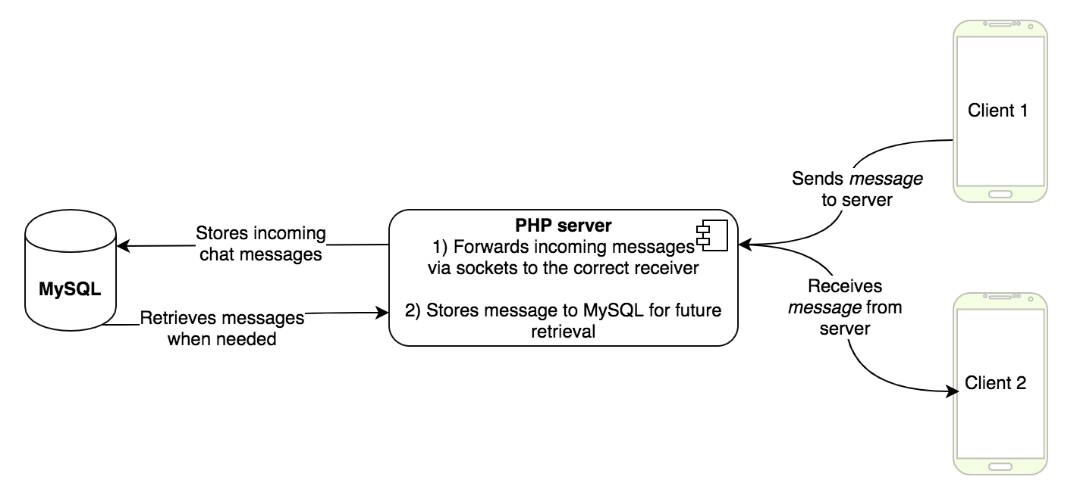


Figure 3.1b

The user of the system provides the port, public key of the receiver, (by selecting which contact to chat with) actually this is the unique number of the receiver of the message. Before the message is sent, the content must be processed.

In this proposed intranet IM system, an intranet based communication system is proposed. It allows android based smartphone and tablet users to send and receive messages over the intranet via Wi-Fi which requires neither any internet connectivity nor any messaging service from the mobile service providers as shown in figure. The motivation is to allow the smartphone and tablet users to communicate in the intranet without paying any internet data charges and avoid having the messages intercepted by a third party. Smartphone users can communicate through the service which is developed and deployed on the intranet server. This service allows intranet users to communicate with each other via Wi -Fi network without using any internet connectivity. When a user wants to send a message to another user, a request goes to the intranet server and now it is the responsibility of the intranet server to deliver the message to the receiving party successfully.

Proposed architecture basically consists of client and server module which may include the following steps:

* First of all, server program (built with PHP laravel framework) runs on server machine.
* Then client program runs on android based mobile device and send a request to connect with server.
* Once the client is successfully connected, the server broadcast the list of all other active users to the client.
* Client can view the list of all active users and can communicate with them.

Intranet server creates a separate connection for each client, for that server creates a separate thread for each client connection. This thread will be responsible to send/receive data to/from the client. When a client sends a message to another client, this message first goes to the server. Then server sends this message to the appropriate receiver. Once the receiver receives the message, it can send. In the same way receiver can reply message to the sender. This application basically uses the concept of socket programming and multithreading. There will be one thread for executing server program and a separate thread to handle each client connection. This approach allows Message transfer between android based devices which is implemented and tested between tablets and mobile phones which uses android platform.

In an intranet system which uses a secure scheme to provide chat services for the users, one of the most important questions is who has the key to decrypt the encrypted message. Taking the new international messaging privacy policy (End to end, E2EE, standard used by Whatsapp, Telegram etc.) into consideration, the requirement to reduce the role of the intranet server and make it zero knowledge of the content of the messages between the user, the best solution is end-to-end encryption, where encryption and decryption are performed at the end nodes. For example, proposed chat application on the mobile phone encrypts the message from the user 1, sends it over to the intranet server, and server sends the encrypted message to the user 2. Now the message can only be decrypted in the local device of the user and the server cannot read the messages, because the server does not have the required keys to decrypt the message.

The proposed secure chat application has the end-to-end encryption (E2EE) method of communication. E2EE means that the end-points are responsible to encrypt data and all encrypted data transferred between different users of the system is only readable to the users who have the key. In this architecture even the intranet server does not have the keys, so even if the intranet server gets compromised or forced by an hacker to reveal the keys of the messages, it will be irrelevant, because the intranet server has little knowledge of the keys and the encrypted messages.

The network architecture of the proposed design can be based on multi-tiered structure in which servers can be physically separated. However, for the sake of this final year project work, all the intranet servers are linked to the intranet service provider and they are stored on a single physical server. The internet service provider server supports two services. The CA (certificate authorization) server; which is the server responsible for certificate issuance. The second server which is IM server is responsible for messaging service which includes offline storage of encapsulated messages in case the recipient is not online (maybe the phone is not switched ON).

Below is the list of security intranet based services which the proposed architecture provides to the user:

* Registration of users
* Secure sessions with the IM and CA servers of the internet service providers
* Keys exchange between users
* Encryption of messages
* Decryption of messages
* Offline secure messages’ storage. This happens when the second party is not online (These offline messages are encrypted and not readable by the server).

**3.2.2 Basic Requirements of the System**

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

1. Providing Session-Level Security (SLS), as a unique key is generated for each session. All the messages exchanged in previous and also future sessions cannot be read by intruder.
2. Each message has its own separate key which brings better security for each single message.
3. All local data on the mobile phone should be encrypted by a separate key derived from the PIN entered by the user during account creation. This PIN never leaves mobile application
4. Support for offline messaging. In case one of the users is not online, the user can still send him/her messages. These messages should be encrypted and stored online, so whenever the user comes online, they can be transferred to the other user for proper replies.
5. All the sessions or information exchanged back and forth between any entities should be encrypted to avoid malicious attacks.
6. The design provides integrity of the messages which assures the recipient that the message has not been modified on the way.
7. The design should provide non-repudiation to assure the message is from the real sender.
8. Not send messages to the local server in clear text form.
9. Never log and save any information regarding any message or its contents or session or event on the server.
10. Not rely on third party servers for message security and handling.

**3.3 Database Design**

The database design model used is quite simple to understand. There are of users. Many users are in many chats with different other users and the other way round. A new message belongs to the user who sent it and to the chat partner where it was posted. Thereby there is the possibility to get all messages of many chats at the same time and also to know who sent it. When the message arrives at the server it is encrypted and stored in the database belonging to the user in the internet service provider’s database. This database also stores the users, chat rooms and all messages. The recipient devices ask the server for the messages, the message is retrieved from the database and sent to the recipient device where it is decrypted and displayed as the new message on the screen. Also a push notification service is implemented so that the recipient is notified for a new message.

REST calls are used for retrieving and posting data to the server. The data is pushed into the database with the server. This server acts like a web service; the chat application can ask for or send data and gets a response from the server. To exchange the data between the chat app and server, it was decided to use JSON. The great advantage of JSON is that its overload is very low (especially compared to XML). The request-JSON is transported to the server via a localhost-POST. The response-JSON is then returned to the app via the localhost-response.

Every request-JSON includes a string parameter called "function". In this parameter one specifies the task of the web service: e.g. return all users- "getusers" or return messages"getmessages" for a given chat room. Also a function that creates resources, e.g. post a message- "postmessage" or create a chat - "createchat". Depending on the function, there have to be other parameters in the request-JSON, for example when using "getmessages" one of the paramters should be the chat\_id. Nearly every function has to know for which user this function is called. This is achieved through the JSON as well.

Depending on the requested function the other parameters of the response-JSON can include e.g. a list of messages in a chat. The data storage on the server side is achieved using the Java Persistence API (JPA) to save all the data on the backend database which is necessary for the web service.

The mobile app’s intranet server is a web app built on a HTML5-based front-end framework which helps to build desktop and mobile web apps. The Web App database communicates with the mobile app by using a RESTful webservice via HTTP.

This webservice is based on a request-response-architecture. This is coupled with a HTML5 WebSocket through which a consistent connection is opened with the backend which stays open over the whole time the client is running.

**3.4 Chat App Use Cases**

**3.4.1 Use Case Actors**

An actor is a user of the system. It includes humans and as well as a process. The actor is the one who initiates the event that leads the use case to begin. The arrow also is an indicator that the interaction starts between the user and the use case. The scope of their action and their overall role in the system defines the set of Use Cases an actor can access. The actors of this chat system are the people chatting with one another.

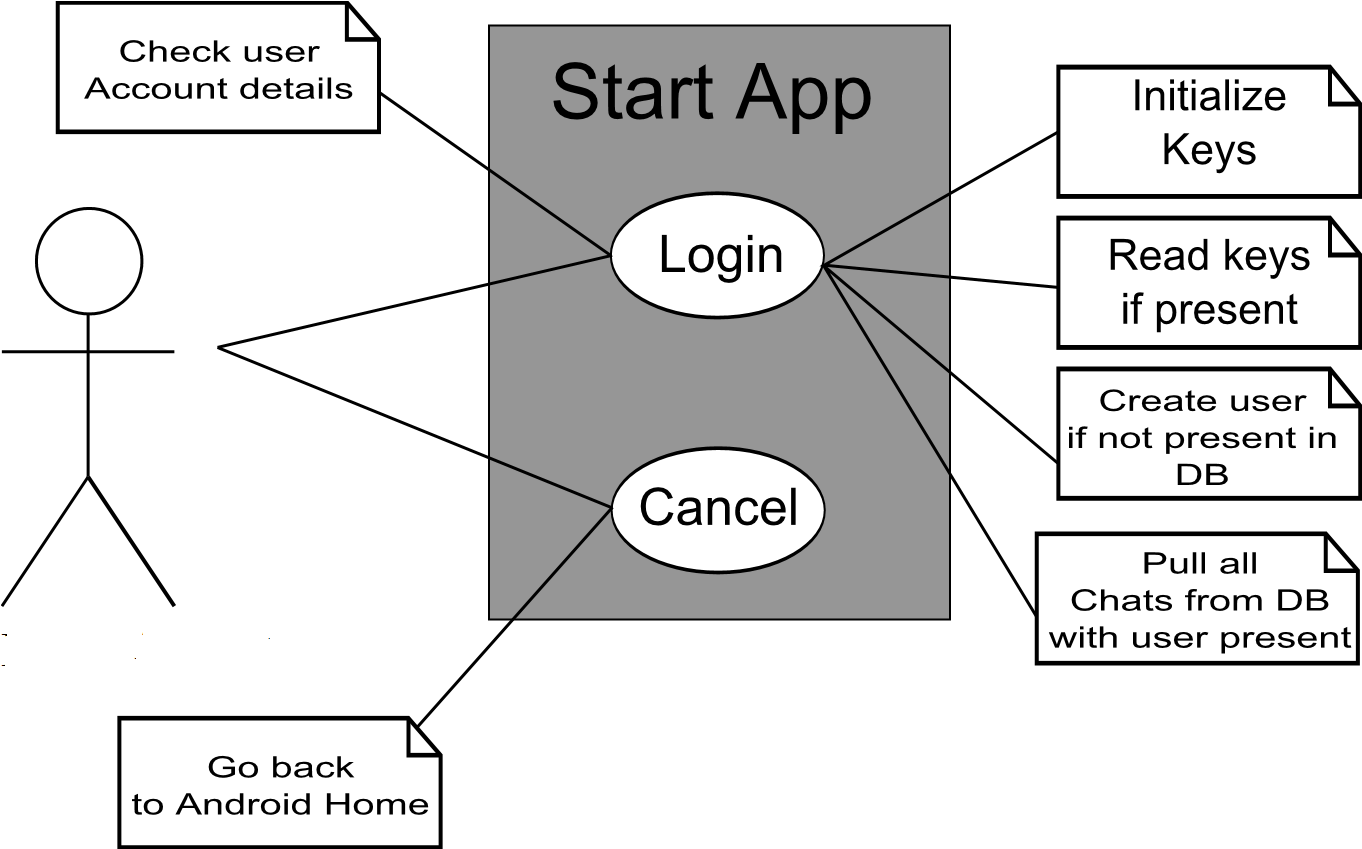
**3.4.2 Use Case Models**

A use case is a single unit of meaningful work/task. It is built with a specific functionality for the proposed system. The keyword ‘Use Case’ can also be used to define a use case. The actors are mainly related to the use cases. Through them the progress of the application could be well known. It describes the basic functionality in a layman’s understanding. It also shows us the requirements of the applications and the constraints associated with it as to what are the functionality a user can access. There are four use cases considered:

* 1. Use Case for Starting the App
  2. Use Case for Sending a Message
  3. Use Case for Receiving a Message
  4. Use Case for Verifying a Public Key

**3.4.2.1 Use Case for Starting the App**

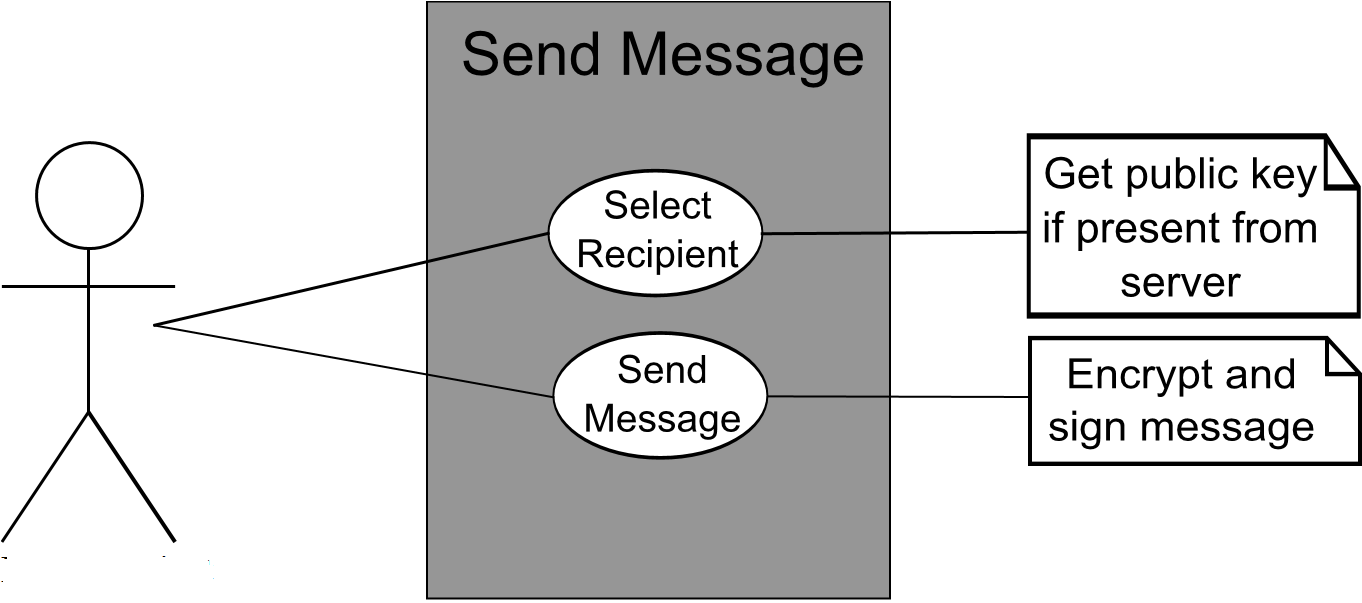
The following Use Case shows the work-flow of the procedure of starting the app login to the account (which invariably generates the key-pairs). The user that is not found on the database is redirected to the signup page to create an account.

Figure 3.2:Use Case for Starting the App

**3.4.2.2 Use Case for Sending a Message**

The following Use Case shows the work-flow of the procedure of sending messages to a recipient. The user selecting the other user to chat with means the user gets a public key if present from the intranet server. Sending a message is preceded by the message being encrypted and signed before it is being sent to the recipient.

To send messages with Java, it requires using the WMA library. The WMA library provides classes to handle messages. It uses the MessageConnection class to initialize a connection to send messages. The format is “message://<reciver\_id>:<port>”, where <port> is the port number which the receiver is listening.

Figure 3.3: Use Case for Sending a Message

**3.4.2.3 Use Case for Receiving a Message**

The following Use Case shows the work-flow of the procedure of receiving messages from a sender. Before the message is displayed to the recipient, the encrypted message is decrypted and the added signature is verified then displayed as clear text to the recipient.

To implement message receiver application, the application must listen to a specified port. As described in Internet Assigned Numbers Authority (IANA), port numbers are divided into three ranges as well-known Ports, registered Ports, and dynamic (Private) Ports. The port number range is from 1 to 65535. The Well Known Ports are those from 0 to 1023 and it can only be used by systems that executed by privileged users. Registered Ports are those from 1024 to 49151, and Dynamic (Private) Ports are those from 49152 to 65535. Both Well Known ports and registered ports cannot be used without IANA registration. Dynamic (Private) Ports can be used in any private applications without registration.

For this application, “50000” have been selected as listening port number which is in the range of Dynamic (Private) Ports. If the port is already used by other application installed on the receiver’s phone the application may not be able to function properly. To receive incoming messages the mobile application must implement the MessageListener class that is standard with J2ME development

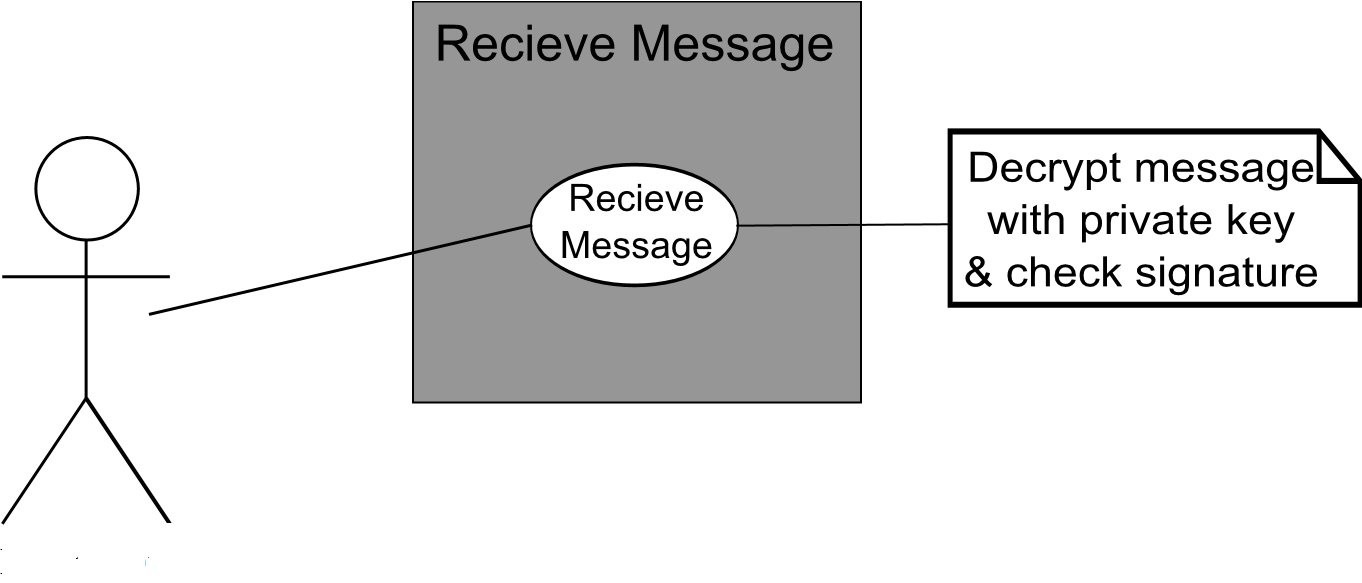


Figure 3.4: Use Case for Receiving a Message

**3.4 Class Descriptions**

The Android chat app is built using native android code. The encryption is like addition of a layer of security to the app. To implement the security criteria, the package com.encryption is created and in it eight classes are created and used, which are: DataHolder, CryptoEncryptedContent, CryptoUtils, CryptoPGP, CryptoPublicKey, CryptoSignedContent, EncDencMessage and NetEncSignMessage as shown in figure 3.5.

* + 1. **DataHolder.java**

The DataHolder class is used to hold public and static variables and data that is shared between the classes. It is done to organize the data in a proper readable manner. The variables it contains are the directory in which the key files are stored, the PGP Private Key, the PGP Public key and a PGP Private Encryption Key. It has a method to read a file and save it as a string, also another method which saves an input stream as astring. The DataHolder class also has methods to test the key function methods.

* + 1. **CryptoUtils.java**

This class is used to initialize or read keys if they are already present. It creates 2 files: a private key file and a public key file with the file extension .asc. If the keys are not found in the app files directory, it can generate a PGP key pair and write the keys which are ASCII armored to files in the app files directory. The private key will be written to file ’secret.asc’ and the Public key will be written to file ’public.asc’. If the keys exist, they are read and verified if their user IDs in the key match the client’s hashId. The ElGamal key is used for encryption while the DSA key is used for signing. There is also a method to delete the keys.

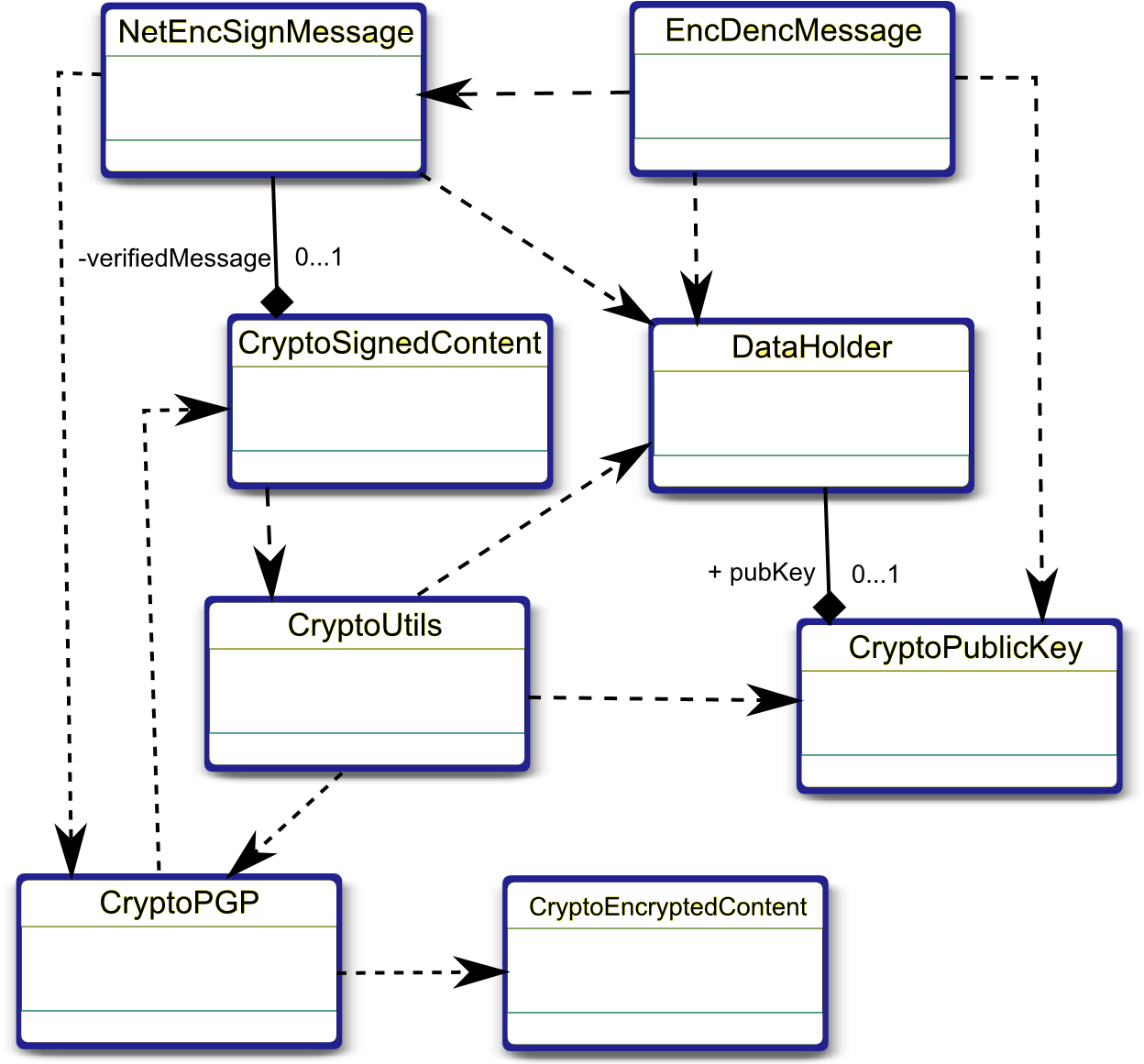


Figure 3.5 Class diagram of the system

* + 1. **CryptoPGP.java**

This class has the technical implementation of signing and encrypting bytes based on the Spongy Castle libraries. This class is where the actual implementation takes place. Strings and data files are converted to a byte stream and then encrypted with the PGP public key ring and the encrypted byte stream can be decrypted and verified with the PGP private key.

* + 1. **CryptoPublicKey.java**

This class is a wrapper class for PGPPublicKeys. The DSA key that can be used verify a signature of the partner and ElGamal key can be used to encrypt data to the partner. The CryptoPublicKey can either be initialized from a file, or from a String containing the ASCIIarmored public key ring.

* + 1. **CryptoSignedContent.java**

This class is a wrapper class for processing raw encrypted messages. After the raw message and public key have been set in constructor, it can decoded and verified through its signature against the public key. This is used for comparing signature against public key if the message has been signed by the owner of the public key. A valid signature will result in the field valid of the class has been set to true.

* + 1. **NetEncSignMessage.java**

This class is a wrapper class for performing the main actions of encryption and a wrapper for decryption. Sign and then encrypt content of raw data string with set keys. After operation, the signed and encrypted content is available in encrypted data. It also has a wrapper method for decrypting and verifying messages.

* + 1. **EncDencMessage.java**

This class is the re-usable wrapper class which has to be called for encryption and decryption. The encryption method takes in a string as input, then using the NetEncSignMessage class method, first the sender private signing key and passphrase is set. The receiver encryption key and passphrase are set. Then the message is signed and encrypted. For decryption, first the recipient private key and passphrase are set and the sender’s signing key is obtained from the public key of the sender. Then the decryption and verification method is called. If any error, then null is returned.

**3.6 Programming Language Used**

It was decided to implement the android chatting system using Java programming language on android studio and the web based intranet server was done using PHP (laravel framework). Among the reasons for picking android for the native mobile are:

* Familiarity;
* Object Orientation;
* Level of abstraction: Java provides a good level of abstraction and it includes rich set of APIs.
* APIs: Java provides messaging APIs in the WMA (Wireless Messaging APIs) set. It also provides a set of APIs to build useful user interfaces.
* Availability: Most mobile phones being sold on the market includes built in Java virtual machine.
* Portability: Java ME makes the application portable on to any java enabled phone. For these reasons java have been chosen as the programming language for the implementations.

In order to develop the IM chatting application, Java Wireless Toolkit development kit was integrated with android studio, android studio was used as development environment. The message sender and receiver application uses the NetBeans Mobility Pack 3.0 to simulate mobile phone environment. Since the mobile application is developed using Java, it should be able to run on any mobile phone that has Java Virtual Machine (JVM).

To enable the application to send and receive chat message on the mobile phone, the J2ME Wireless Message API (WMA) library was used. The WMA library has various classes and methods to construct a message. In order to develop the security of chat, the Bouncy Castle Crypto package was used. Bouncy Castle offers lightweight API for many encryption procedures. These API simplify the task in building the security solution in Java mobile application. The package contains a lightweight API suitable for use in many environments including J2ME. Bouncy Castle Crypto libraries that are only important for security algorithms were added. The selection of adding only necessary libraries of Bouncy castle package is made in order to avoiding memory overhead of the devices. The following subsections describe the implementation of each part of the chatting application.

**3.7 Procedure Phases and Time Frame**

The implementation of the project was divided into the following phases:Requirement phase, Analysis Phase, Design phase, Implementation and testing phase:

* Requirement phase: here, a complete description of the behavior of the system was done. The interaction users would have with the software was clearly mapped out for clear development (coding).
* Analysis Phase: Formal enquiry was carried out in order to identify a better course of action to develop the system, thoughts were shared, and online resources were perused for clear information.
* Design phase: The Graphical User Interface for the mobile application was developed. This was done using android studio (Java programming language).
* Implementation Phase: This is the part of the process where the software engineering of the information system was actually done. The coding from scratch using Java on android studio and SQLite. Scripts for different modules were written and fully tested okay. This phase is in progress.

The project was completed in a six month timeline (first and second semesters). It is divided into two segments completed in each semester. The first segment is the gathering of relevant information and other resources for the successful completion of the project, this entailed the preparation of the chapter one (introduction) and two (review of literatures on past related works) of the project report. The second segment involved the actual implementation and discussion of the results attained.