KANTIPUR ENGINEERING COLLEGE

(Affiliated to Tribhuvan University)

Dhapakhel, Lalitpur

[Subject Code: CT755]

A MAJOR PROJECT REPORT ON

END-TO-END ENCRYPTION CHAT APP

Submitted by:

Anup Chaudhary	**[38659]**
Chris Gurung	**[38672]**
Himalaya Pal	**[38678]**
Kundan Giri	**[38679]**

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

April, 2023

END-TO-END ENCRYPTION CHAT APP

Submitted by:

Anup Chaudhary	**[38659]**
Chris Gurung	**[38672]**
Himalaya Pal	**[38678]**
Kundan Giri	**[38679]**

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

Kantipur Engineering College

Dhapakhel, Lalitpur

April, 2023

COPYRIGHT

The author has agreed that the library, Kantipur Engineering Collage, may make this report freely available for inspection. Moreover the author has agreed that permission for extensive copying of this report for scholarly purpose may be granted by the supervisor(s), who supervised the project work recorded herein or, in their absence, by the Head of the Department wherein this project was done. It is understood that due recognition will be given to the author of this report and to the Department of Computer and Electronics Engineering, Kantipur Engineering College in any use of the material of this report. Copying or publication or other use of this report for financial gain without approval of the Department of Computer and Electronics Engineering, Kantipur Engineering College and author's written permission is prohibited.

Request for permission to copy or to make any other use of the material in this report in whole or in part should be addressed to:

Head
Department of Computer and Electronics Engineering
Kantipur Engineering College
Dhapakhel, Lalitpur
Nepal

KANTIPUR ENGINEERING COLLEGE
DEPARTMENT OF COMPUTER AND ELECTRONICS ENGINEERING

APPROVAL LETTER

The undersigned certify that they have read and recommended to the Institute of Engineering for acceptance, a project report entitled ”End-To-End Encryption Chat app ” submitted by

Anup Chaudhary	[38659]
Chris Gurung	[38672]
Himalaya Pal	[38678]
Kundan Giri	[38679]

in partial fulfillment for the degree of Bachelor in Computer Engineering.

....
Supervisor
Er. Bikash Shrestha
Sr. Telecom Engineer,
Nepal Telecom

....
External Examiner
External's Name
External's Designation
Second Line of Designation (if required)

....
Er. Rabindra Khati
Head of Department
Department of Computer and Electronics Engineering

Date: April 23, 2023

ABSTRACT

Data security is a crucial concern that ought to be managed to help protect vital data. Cryptography is one of the conventional approaches for securing data and is generally considered a fundamental data security component that provides privacy, integrity, confidentiality, and authentication.

In the current world where communication has been made easy such that you could talk to a person on the other side of the world with a press of the button. With the increase in availability of internet service you can send texts, photos ,files through the internet in a matter of seconds and for far less cheaper. This is achieved through different chat applications. With the increased usage of such chat applications the contents of such messages contains more that just simple messages to friends and families but also very important information and files which on the wrong hands could cause a huge catastrophe. As such End-to-End security is needed to safely exchange private information with each other without worrying about data. With this project we aim to provide an End-to-End encrypted chat apps.

This project approach End-to-End encryption method to provide secure communication that prevents third parties from accessing data while it's transferred from one end system or device to another. The End-to-End encryption method is implemented using the Asymmetric key encryption algorithm (RSA). We used RSA to encrypt that encoded data. In this way we can maintain the data more securely.

Keywords – RSA

ACKNOWLEDGMENT

We would like to acknowledge Er. Bikash Shrestha, our project supervisor who is providing us guidence and helping us with different views of application, development and scalability as well as helped us in development of our project. His guidence has been corner stone in developing chat app. We are able to accomplish this project with his valuable direction and suggestions.

We would like to express our notable thanks to Department of Computer and Electronics Engineering for providing us environment to dig the latest technology, investigate it and do research on those areas as Major project. We are also grateful to Er. Rabindra Khati, HOD, Department of Computer and Electronics Engineering for presenting useful suggestions regarding the queries put forward regarding the project.

Our special appreciation goes to Lecturers of Kantipur Engineering College for guiding us in this project and helping to better cope with the difficulties along with giving supportive hands. Without their valuable guidence it would have been difficult for us to come to a conclusion to work with this technology.

Last but not least, we would appreciate all our seniors and friends for their support and constant inspiration which helped us to withstand in our success in this project. to display members name under Acknowledgement

Anup Chaudhary	[38659]
Chris Gurung	[38672]
Himalaya Pal	[38678]
Kundan Giri	[38679]

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Background

Digital communications surveillance is a major security concern in the world at large. End-to-End (E2E) encryption in mobile communication applications delivers confidentiality between users, defending messages against snooping. Several widespread communication tools (WhatsApp, Signal, Telegram, iMessage) have implemented end-to-end encryption, as a major selling point. Yet, the understand of the security goals (confidentiality, integrity, authentication) remain vague to users, such as how the security goals offer protection, and if they value that protection.

With the growing use of the internet as a medium for communication, it becomes imperative to secure personal and business' online communications. Consequently, the motivation to execute attacks increases and preventing against these attacks using technology that is almost unbreakable was considered. This technology, if employed accurately, could avert large-scale attacks. End-To-End Encryption suggests a maintainable answer to the continuing challenges of internet security . End-to-end encryption describes the process of secure exchange of data from sender to recipient; preventing third-parties from accessing the data during transmission. All information is encrypted by the sender and the recipient decrypts it. During transmission, the content is completely encrypted, which means that no third parties can access or tamper with it during transmission . Several cryptographic algorithms are used alone or combined for the encryption purposes.

The components of End-To-End Encryption includes : The identity component authenticates users. The protocols component handles the key exchange and the algorithm. The algorithm uses scientific process to encrypt the data, and it cannot be decrypted without the predetermined key. Secure implementation and operation ensure the End-To-End Encryption process is not vulnerable to attacks on the hardware side. These components work together to deliver a system that operates efficiently to offer the best security to end users.

Internet security is an extremely vital issue in computer science, due to the increasing acceptance of online communication. Since e-mailing services became public, questions arose about how secure they are. Furthermore, the need to secure the internet was made popular by shopping, banking, and other financial transactions via the internet . Is End-To-End Encryption the best technology to ensuring data security? In recent times, most of the communications between clients and servers are secured using Transport Layer Security. However, communications between clients are not yet secured . With plaintext communication vulnerable to hackers . Consequently, security researchers have proffered the usage of End-To-End Encryption.

This research focused on providing a performance evaluation experiment on RSA algorithm and its usage in end to end encrypted chat app. RSA is based on number theory, using two prime numbers or mathematical operation to randomly produce the public and private keys. The public key (which is public) is used for encryption, and the private key (which is private) is used for decryption. Sender encrypts the communication using public key of the recipient and when the communication is received, the recipient can decrypt it with its private key.[1]

1.2 Problem statement

Traditional messaging like SMS are very unsecure as it travels through the network in plain text. Since messages might contain very sensitive informations the messages shouldn't be accessible to any other person beside the actual people involved in the conversation. Furthermore in unsecure messaging any third party can pretend to be the intended person and gain access or send wrong information.

1.3 Objectives

The application aims to provide data security in communication system. The application focuses on simplicity of design, having user-friendly interface and to be easily understood.

The main objectives of the application can be enumerated as follows:

- To provide end-to-end text message security.
- To provide platform for sending messages from one person to another.

1.4 Applications

The application is an online web application. This system can be used to provide end to end encryption of data and also used to provide secure connection between user between users with smooth and clean UI.

1.5 Project features

The application, is targeted towards the general population, so the core features of this applications can be listed below:

- We can send messages
- It provides end-to-end data security in communication system.
- Simple and ease for general population

1.6 System Requirements

1.6.1 Software Requirement

Application is targeted towards a general market, so it is aimed to be fully optimized enough for any low-range to high-range systems, so listed below are the software requirements for the development and operation of this system:

- Operating System: Windows 8 or above
- Browsers: Google Chrome, Firefox, etc
- Mongodb, Node.js Server
- React js

1.6.2 Hardware Requirement

Hardware configuration and requirements for the operation of this application are as follows:

- Intel Core 2 Duo Processor (Recommended i-series processors or more) with minimum of 2GB RAM for application operation
- Server with optimum node speed

CHAPTER 2
LITERATURE REVIEW

2.1 Research on End-to-End Encryption chat app

There are currently millions of monthly active users worldwide of different chat applications currently. There are two types of architecture in those applications, client-server and peer-to-peer networks. In a peer-to-peer network, there is no central server and each user has his/her own data storage. On the contrary, there are dedicated servers and clients in a client-server network and the data is stored on a central server. Security and privacy in chat applications have a paramount importance but few people take it seriously. In a test done by the Electronic Frontier Foundation, most of the popular messaging applications failed to meet most security standards. These applications might be using the conversations as an information for certain purposes. Moreover, reading the private conversations is certainly unacceptable in terms of privacy. Most applications only used Transport Layer Security (TLS) for securing channel, the service provider has full access to every message exchanged through their infrastructure . Therefore, these messages can be accessed by attackers. Therefore to maintain protection and privacy, messages should be encrypted from sender to receiver and no one can read messages even the service provider, in addition to protecting the local storage of the device.

There are different Encryption algorithms that can be utilized to provide secure messaging environment. Thus, messages will circulate as encrypted form in transmission medium, not as clear text. Somebody who has seized encrypted data does not obtain original message from the encrypted data unless they possess the necessary method or a key. Encryption methods are divided into the following categories: private key cryptography and public key cryptography.

In a symmetric key algorithm, the sender and receiver must have a shared key set up in advance and keep secret from all other parties; the sender uses this key for encryption, and the receiver uses the same key for decryption. In this case, except for transmitted encrypted message, encryption key must also be submitted confidentially, which is one of the disadvantages of private-key cryptography. If a third person who has managed to enter the system operator or listen to transmission medium seizes the key value, s/he can turn the encrypted data into original data. The most important feature of public key cryptography which is another method is that the key value used to encrypt the message is different from the key value used to decrypt the message. Each user has two keys in this method: public key and private key. The public key of the user can be viewed by anyone. The private key is kept secret by the user. When someone wants to send a message to user, they use the user's public key and create the encrypted message and then send the encrypted data to user. The user decrypts the encrypted data with her/his private key and obtains a meaningful message.[2]

2.2 Research on Client / Server Cryptography-Based Secure Messaging System Using RSA Algorithm

In today's world, computer networking has become an integral part of life. There are many different networks available to share information between groups of devices through a shared communication medium . They are mainly differentiated by the physical medium and protocol standards. Ethernet is a prime wired networking standard which is an obvious choice for many network applications due to reliability, efficiency, and speed. Ethernet standard is used in various application segments . Figure below shows the Client/Server model architecture that has been used in most network systems and in this study specially.

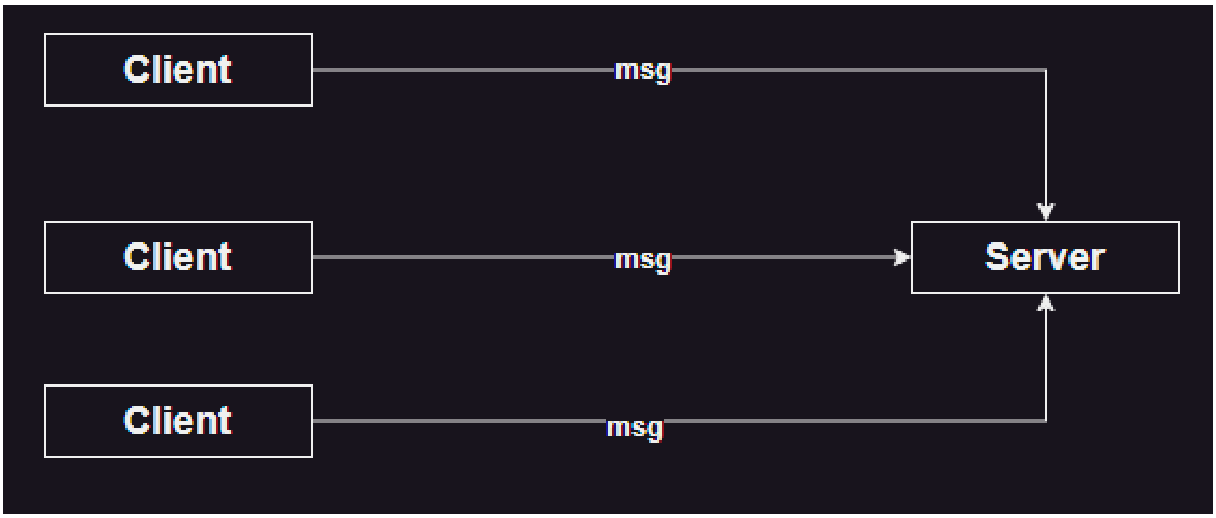

Figure 2.1: A Client/Server architecture

The client side could be any type of smart devices (desktop, laptop, smart phone, etc.). The server part is one device that control and pass messages and opining the connections among clients and/or between clients and server . The Internet part could be one device to isolate the network overall into two main parts: client(s) and server, it could be a switch or hub or router or just a cable.

A very important aspect in the world of software development is the security of data that flows through open communication channels. In our web applications, there is an intensive exchange of data via different protocols, like http, between client applications which presented as browser applications and server side applications. The importance and confidentiality of data may be different depending on the specifics of the web application, and the possibility of interception by a third party increases with perfection of hacking techniques in the world of IT. What can be done to prevent access to the data by your traffic listener? If we exchange with data between the client applications and server we don't want the information to be stored as open text on the server, which will be accessible in case of server crack.

Every day people used chat area, through the users (clients) scan chat or send messages to selected users. However, the security components in chat area application are to make sure all information from clients is protected from hackers. The chat messages from users can easily transform by expert hackers, without a good enough security components. In this way, a chat area interface (CAI) is required technique to secure a chat message from hackers. The cryptography is significant to keep private data secure and to avoid unauthorized access.

Basically, the proposed messaging/chat system is expected to provide a communication channel between clients via a server using encryption based on RSA in a Client/Server environment. The goal for this study is to use client/server architecture to accomplish secure chat between clients. All the used encryption processes based on RSA algorithm.

The very term client-server was initially applied to the software architecture, which described the distribution of the execution process by the principle of interaction of two software processes, one of which in this model was called the client and the other the server. The client process requested some services, and the server process ensured their execution. It was assumed that one server process can serve a lot of client processes. One of the client/server application is that “chatting”. Chatting alludes to one kind of correspondence over the Internet that offers a continuous transmission of instant messages from sender to beneficiary or over a server that is control and deal with the gatherings (customers) to convey.[3]

2.2.1 Client/Server

The used client/server model describes how a server provides resources and services to one or more clients. Examples of servers including web servers, chat servers, and file servers. Each of these servers provide resources to client devices. Most servers have a one-to-many relationship with clients, meaning a single server can provide resources to m Computers. In order to meet the main requirements of businesses, networks themselves are becoming quite complex multiple clients at one time .[3]

2.2.2 Chat Service

A secure chat service provides the ability to have real time secure discussions among users electronically, one-to-one or in groups session . A public network accumulates information slightly, rather than on a user’s individual computer that is used to keep in touch with people.[3] A secure chatting between client and server to make a safe and reliable communication, the benefits are :

- Allows for instant communications between users
- Uses real time chat over the network that can eliminate costly long distance charges.
- Allows for rapid query and rapid responses.

While the negative points of chat service can be listed as following:

- Security problems of instant messaging program
- Secure chats in most cases are routed through a server system, where the service is provided and that is a single point where all messages can be intercepted.
- Chat programs can provide an open avenue of attack for hackers, crackers, spies

and thieves.

2.2.3 RSA Encryption

In this, an encrypted chat program designed to ensure a safe mode of communication between two users. It uses RSA encryption to encode and decode messages in a terminal window. RSA is widely used public-key cryptograph and authentication system for data encryption of digital messaging transactions such as e-mail over the intranet, extranet and Internet. Clients exchange public keys and encrypt outgoing text with the intended recipient's public key. Each user connects to a central server which forwards messages to the intended recipient. On the receiving end, the program utilizes a client's private key to decrypt received messages. In 1977, Ron Rivest, Adi Shamir and Leonard Adleman introduced a cryptographic algorithm, RSA, which is named for the first letter in each of its inventors' last name. RSA's motivation is DiffieHellman Algorithm which describes the idea of such an algorithm that enables public-key cryptosystem.
Encryption algorithm is deployed to encrypt messages exchanged with the proposed chat gateway. The chat messaging environment showed a great potential to host real-time interactive interaction system which is supported by RSA encryption methodology to preserve the security of the message stream.
Choosing the key size in RSA encryption is of great importance. As the size of the key increases, the security level of the system, the complexity and the resistance of encrypted text increases . These advantages make it difficult to decrypt ciphertexts and break passwords. However, in addition to these advantages, the encryption key creation time, text encryption time, and mobile device RAM consumption increase . These disadvantages are factors that will influence the effective use of the application. For this reason, the advantages and disadvantages of key dimensions should be determined and the most suitable key size should be preferred. [3]
Currently, RSA is secure using 2048 and 4096-bit key lengths. Larger keys will be required in a near future.Theoretically, RSA keys that are 2048 bits long should be good until 2030. If so, isn't it a bit early to start using the 4096-bit keys that have become increasingly available in encryption-enabled applications? According to " NIST Special Publication 800-57 Part1"2048-bit RSA keys are roughly equivalent to a Security

Strength of 112. Security strength is simply a number associated with the amount of work required to break a cryptographic algorithm. Basically, the higher that number, the greater the amount of work required. It implies longer keys are more difficult to break and are hence more secure.

Security Strength	RSA Key length
$<= 80$	1024
112	2048
128	3072
192	7680
256	15360

According to that publication, 112 security strength (which corresponds to 2048-bit keys) is considered to be acceptable until 2030. Again, here's a portion of that table for reference.

Security Strength	Through 2030	2030 and beyond
< 112	Disallowed	Disallowed
112	Acceptable	Disallowed
128	Acceptable	Acceptable
192	Acceptable	Acceptable
256	Acceptable	Acceptable

So now we know 2048 bit keys are indeed acceptable until 2030 as per NIST. So where does that put our 4096 bit keys? Incidentally, the document is silent about this particular key length. However, because the two tables indicate that 3072-bit keys (whose security strength is 128) and 7680-bit keys (whose security strength is 192) are good beyond 2030, we can safely say 4096 bit keys (which are somewhere in between) should likewise be considered secure enough then.

In fact, since 2048-bit keys are supposed to be disallowed after 2030, we know for certain that 4096 bit keys are going to be more suitable in production environments than 2048 keys when that time comes. But since we're still at least a decade away from 2030, it's probably not yet necessary to migrate from 2048 to 4096, right? So why then

are we already seeing options for 4096-bit keys in some security applications?
Well, there is couple of reasons. One is simply to make the application future proof. A future proof security solution can mitigate the risk of cyber threats. We know that cyber criminals are always one step ahead of security professionals, so we're not 100percent sure 2048-bit keys are going to remain unbreakable before 2030. [4]

In RSA, a sender encrypts a message using the recipient's public key, and the recipient decrypts the message using their private key. sWhen transferring an RSA public key, it is important to ensure that the key cannot be easily intercepted and replaced by an attacker. One common technique for protecting the key during transfer is to use salting, also known as padding or randomization.

Salting involves adding random data to the beginning or end of the public key before transferring it. This random data, or salt, makes it more difficult for an attacker to intercept and replace the key with their own. The recipient of the key can then strip off the salt before using the key for encryption or decryption.

In summary, salting is a simple and effective technique for protecting RSA public keys during transfer. It adds an extra layer of security to the key exchange process and makes it more difficult for attackers to intercept and replace the key.

2.3 Existing system:

In this section we briefly introduce many of the popular chat applications. Some of these applications are not public or open source so it is difficult for these to get evaluated by the developer's community, security experts or research academic.

2.3.1 Viber

Viber is an instant messaging and Voice over IP (VoIP) application for smartphones developed by Viber Media. In addition to instant messaging, users can exchange images, video and audio media messages. Viber recently supported the end-to-end encryption to their service, but only for one-to-one and group conversations in which all participants are using the latest Viber version 6.0 for Android, iOS or Windows 10. At this time, in the Viber iOS application for iPhone and iPad, attachments such as images and videos which are sent via the iOS Share Extension does not support end-to-end encryption. Viber has privacy issues such as adding a friend without his knowledge or adding him to a group without his permission. Plus that, local storage is not secured. It is not open source making it difficult to evaluation.[2]

2.3.2 Whatsapp

WhatsApp is one of the most popular messaging application, recently enabled end-to-end encryption for its 1 billion users across all platforms. WhatsApp uses part of a security protocol developed by Open Whisper System, so provides a security-verification code that can share with a contact to ensure that the conversation is encrypted. It is difficult to trust in WhatsApp application completely because the application is not open source, making it difficult to verify the functioning process and match them with the work of the encryption protocol which was announced. .[2]

2.3.3 Telegram

Telegram is an open source instant messaging service enables users to send messages, photos, videos, stickers and files. Telegram provides two modes of messaging is regular chat and secret chat. Regular chat is client-server based on cloud-based messaging, it does not provide end-to-end encryption, stores all messages on its servers and synchronizes with all user devices. More, local storage is not encrypted by default. Secret chat is client-client provides end-to-end encryption. Contrary to regular chat messages, messages that are sent in a secret chat can only be accessed on the device that has been initiated a secret chat and the device that has been accepted a secret chat they cannot be accessed on other devices. Messages sent within secret chats can be deleted. at any time and can optionally self-destruct. Telegram uses its own cryptographic protocol MTProto which is based on 256-bit symmetric AES encryption, 2048-bit RSA encryption, and Diffie–Hellman secure key exchange, and has been criticized by a significant part of the cryptographic community about its security. The registration process of Telegram, Viber and WhatsApp depend on SMS. SMS is transported via Signaling System 7 (SS7) protocol. The vulnerability lies in SS7. Attackers exploited SS7 protocol to login into victim's account by intercepting SMS messages. Because of Telegram cloud-based, the attacker exploits it and makes full control of the victim account and can prevent him to enter into his account. To make the account more secure should activate two-factor authentication [2]

2.3.4 Facebook Messenger

Facebook Messenger is a popular messaging service available for Android and iOS. It provides two modes of messaging one is regular chat and another is secret conversations. Regular chat does not provide end-to-end encryption only secure communication by using TLS, and it stores all messages on its servers. Secret conversations have the same idea of Telegram secret chat . [2]

2.4 The Challenge- Making it real-time

For any app to feel real-time, the user needs to be kept updated with any activity happening as soon as possible. The challenge arises in selecting and implementing a suitable development technique. With the traditional request-response model, we have few options:

2.4.1 Refresh Webpage

The user might refresh the web page time-to-time to check for message updates. But that is not an optimal solution. This may result in bad UX.

2.4.2 HTTP Protocol

The concept of HTTP request-response is widely used. But this requires establishing a TCP connection every time data is sent to the server. Being a one-way synchronous communication protocol, this may result in a lot of overheads while creating and destroying a TCP connection every time a message is sent in real-time chat applications.

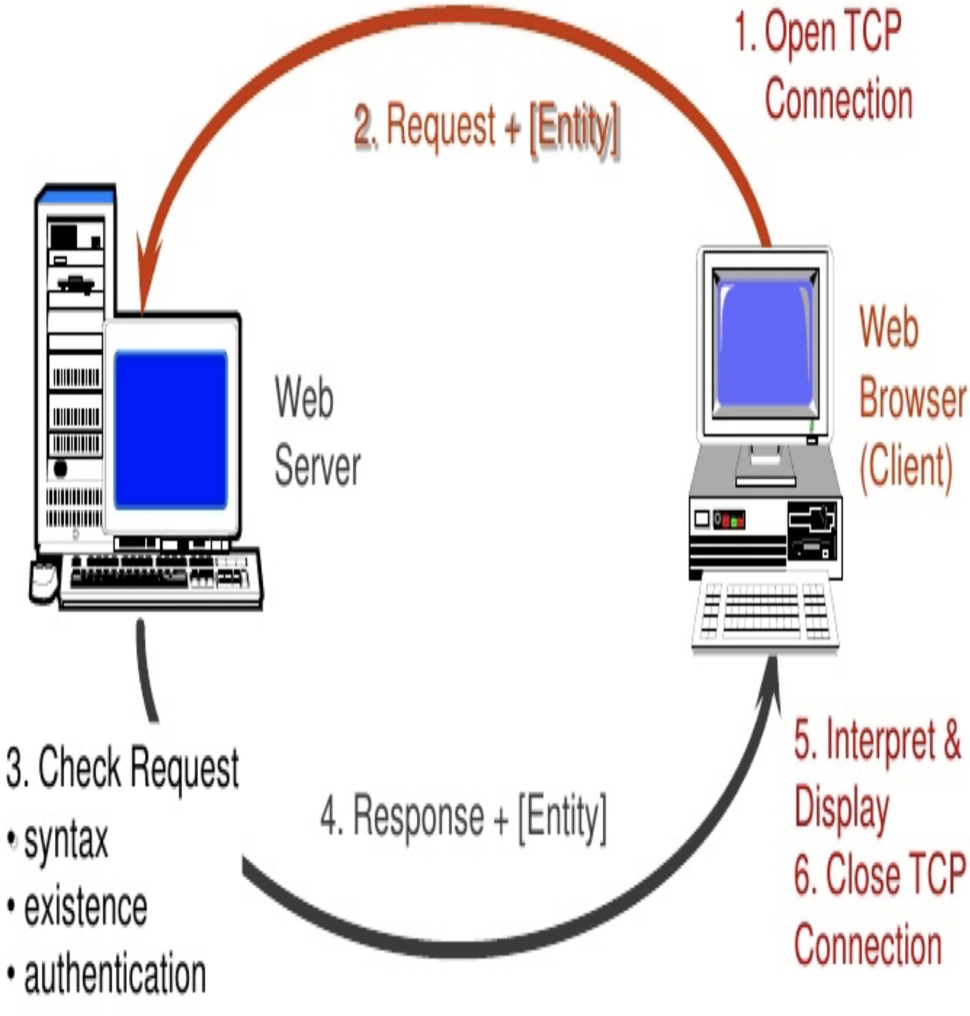

Figure 2.2: Http Request-Response Cycle

2.5 Web Sockets

This concept resolves most of the issues we just discussed. It implements instant two-way communication of messages with a persistent connection just as required for developing a real-time system. NodeJS offers several libraries to implement this technology. What we will be utilizing for our application is the Web Socket API with 'Socket.io' library.

2.5.1 Socket.io Api

Socket.IO is a library that enables low-latency, bidirectional and event-based communication between a client and a server. It is built on top of the WebSocket protocol and provides additional guarantees like fallback to HTTP long-polling or automatic reconnection.

2.5.2 Socket.io Handshake

The handshake in Socket.IO is like any other information technology related handshake. It is the process of negotiation, which in Socket. IO's case, decides whether a client may connect, and if not, denies the connection.

2.5.3 Exchanging Messages

- The messages are exchanged in the form of data frames rather than a stream of data
- It is a bi-directional flow of data
- The event listeners of the ws object are used for message exchange
- The closing handshake can take place either by the client or the server. Reconnection has to be done manually.

CHAPTER 3
SYSTEM ANALYSIS AND DESIGN

3.1 Functional Requirement

- User Registration and Authentication: The app should allow users to create accounts, log in, and maintain user profiles securely.
- Chat Messaging: Users should be able to send and receive text messages in real-time or near real-time.
- Group Chats: The app should support group conversations where multiple users can participate simultaneously.
- Multimedia Messaging: Users should be able to send and receive various types of media, such as images, videos, and files.
- Message Notifications: Users should receive notifications for new messages, either through push notifications or in-app alerts.
- Message Status: Users should be able to see the status of their messages, such as delivered, read, or typing indicators.
- Message History: The app should store and retrieve chat history, allowing users to view past conversations.
- Search Functionality: Users should be able to search for specific messages or users within the app.
- Integration with External Services: The app may need to integrate with external services like cloud storage or third-party authentication providers.

Till now we have developed user registration and authentication, messaging requirements, group chats, searching.

3.2 Non-Functional Requirements

- Performance: The app should be responsive and provide fast message delivery and retrieval, even under heavy load.
- Scalability: The system should be able to handle a growing number of users and messages without significant performance degradation.
- Reliability: The app should be highly reliable, ensuring message delivery and

minimizing downtime or service disruptions.

- Security: The app should implement strong security measures, including data encryption, secure authentication, and protection against common vulnerabilities.
- Usability: The app should have a user-friendly interface, intuitive navigation, and support for multiple platforms (e.g., mobile and web).
- Availability: The app should be available and accessible to users whenever they need it, with minimal planned maintenance periods.
- Compatibility: The app should work seamlessly across different devices, operating systems, and web browsers.
- Privacy: The app should respect user privacy by handling personal data appropriately, following relevant regulations and guidelines.
- Backup and Recovery: The app should have mechanisms in place to back up data and enable recovery in case of data loss or system failures.

3.3 Feasibility Analysis

Feasibility is a measure of how benificial or practical the development of a system will be to an organization or enterprise. In other words, a feasibility study is an evaluation and analysis of the potential of the proposed project which is based on extensive investigation and reasearch to give full comfort to the decision masker. A system should have a responsible cost and should be technically and operationally feasible to be actually implemented. The two criteria to judge feasibilty are cost required and value to be attained. The feasibility of ”End to End encryption Chatapp ” is analyzed under the following four headings:

3.3.1 Economic Feasibility

Economic Feasibility is the measure of the cost effectiveness of the project or the solution. This is often called a cost benifit analysis.

Based on our economic analysis for development and operational cost, the system is being developed and operated economically. For development, the required devices are readily available, so it is feasible. Also, it is economically feasible to the consumers as it costs no charge to use the platform.

3.3.2 Schedule Feasibility

Based on the objectives and the time left for the development. The schedule is found to be feasible.

3.3.3 Technical Feasibility

Technical Feasibility is the measure of the practically of a specific technical solution and the availability of technical resources and expertise.

Technically, the system is feasible enough and easy-to-use for both technical and non-technical groups of people. It provides a user-friendly environment along with features using the latest technologies. The system provides a layout most of the applications people are used to anyway so it will be easy to use.

3.3.4 Operational Feasibility

Operational feasibility is the measure of how a well specific solution will work in the organization. It is also measure of how people feel about the system of the project. It also analyze the inside operation on how a deemed process will work, be implemented, and dealing with changes resistance and acceptance.

For the operation of the system, the person does not need to excel in using a computer. Since, the event may not always be related to the technical fields, someone with minimum knowledge about computer and technology can also get benefit from the system. Similarly, one can get access to the system as a web-based application. There is no requirement of huge and expensive hardware.

3.4 Use Case Diagram:

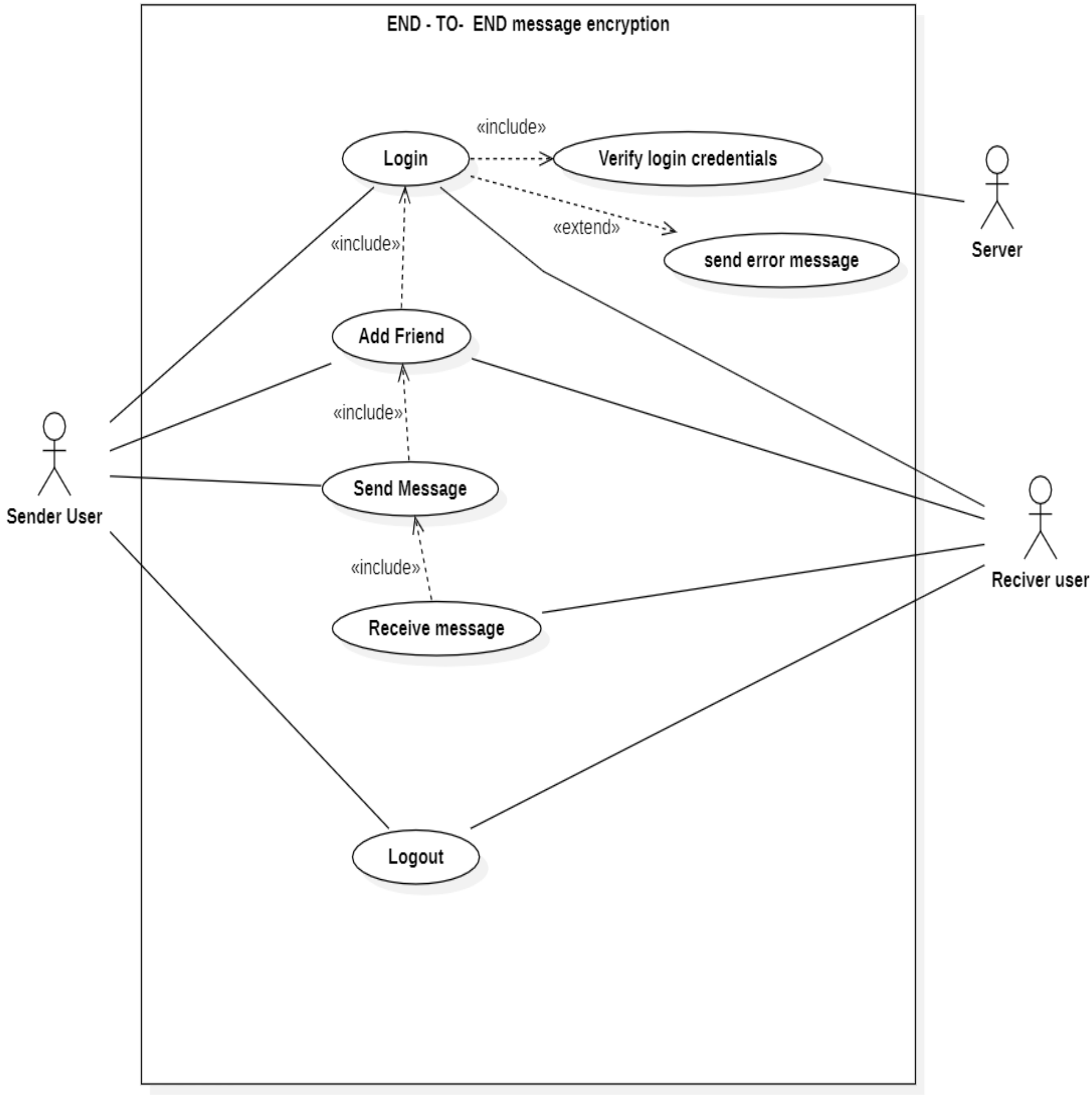

Figure 3.1: Use Case Diagram

3.5 Data FLow Diagram

3.5.1 DFD level 0

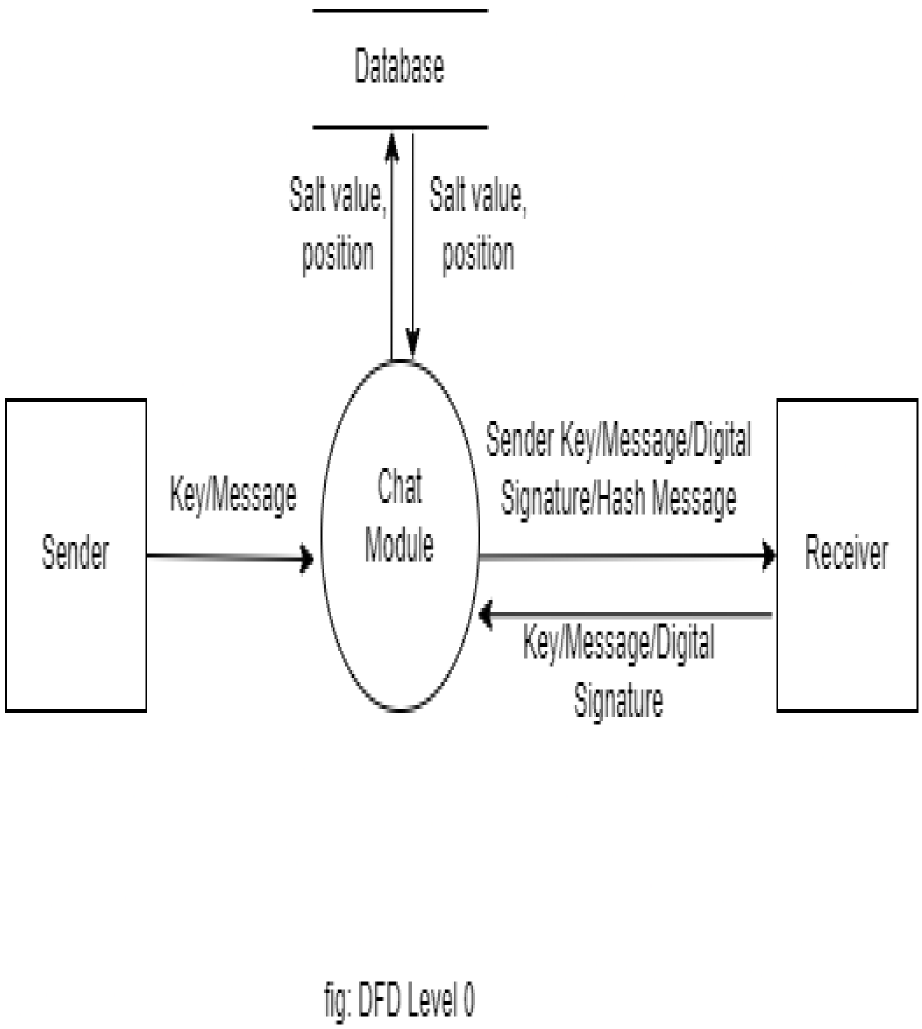

Figure 3.2: DFD level 0

3.5.2 DFD level 1

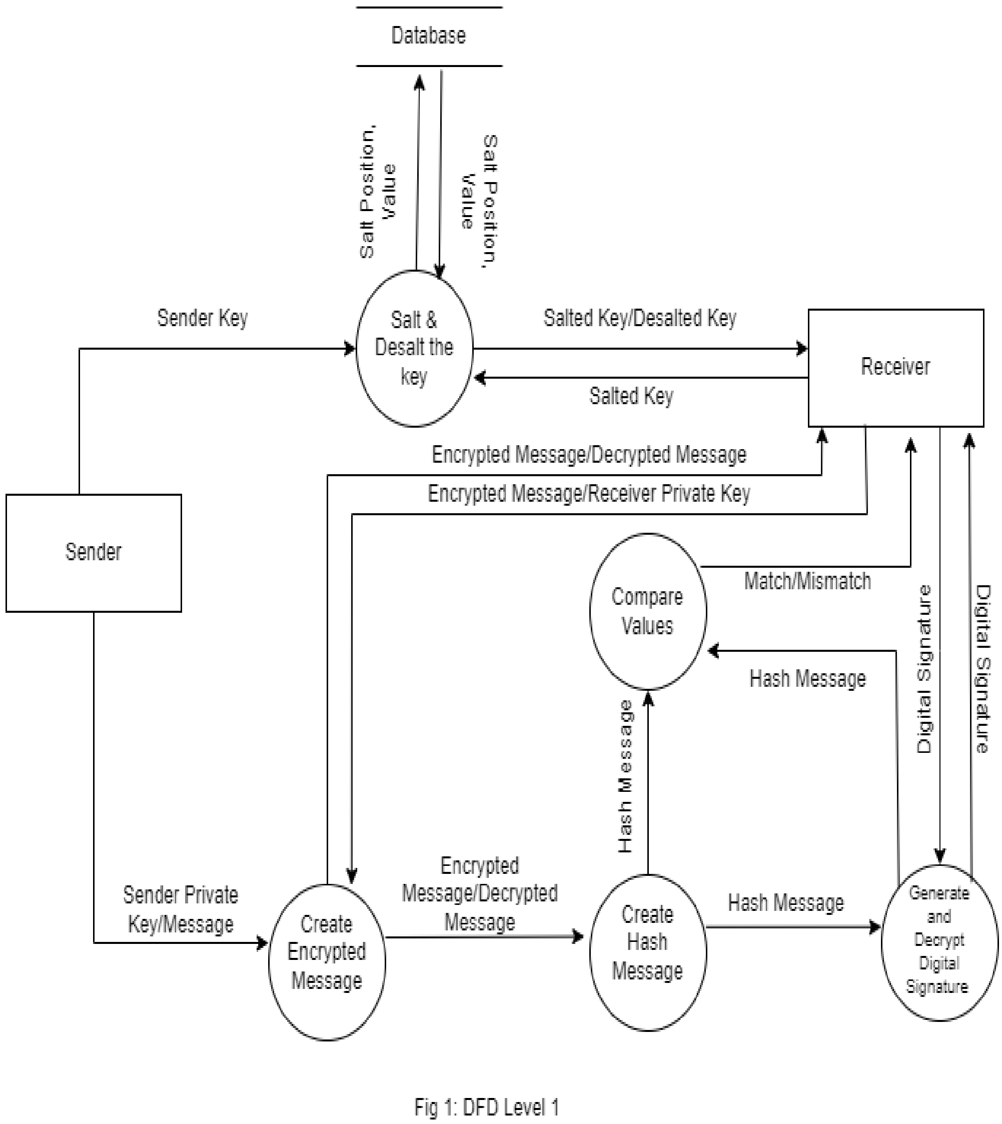

Figure 3.3: DFD level 1

3.6 Sequence diagram

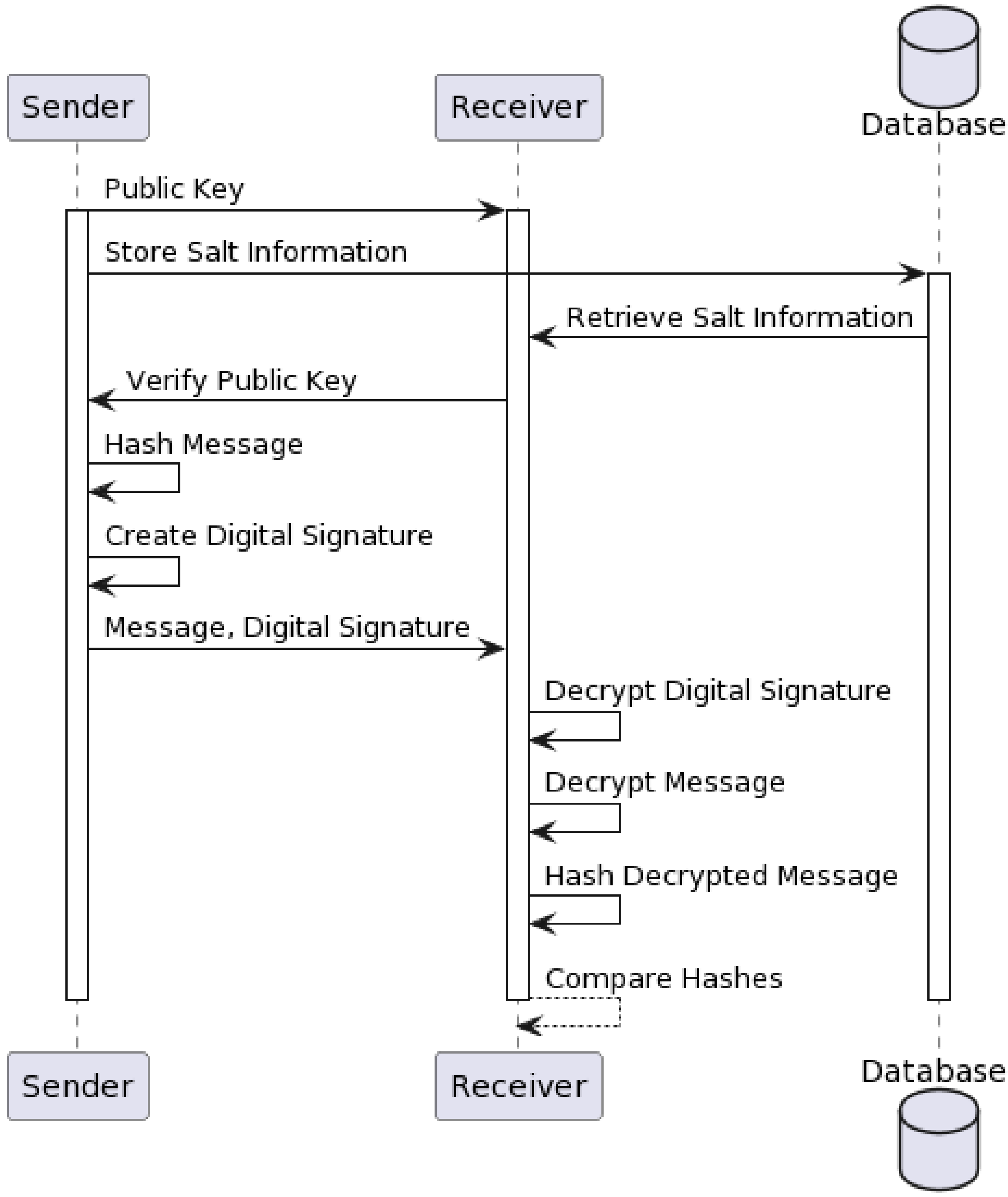

Figure 3.4: Sequence diagram

3.7 Activity diagram

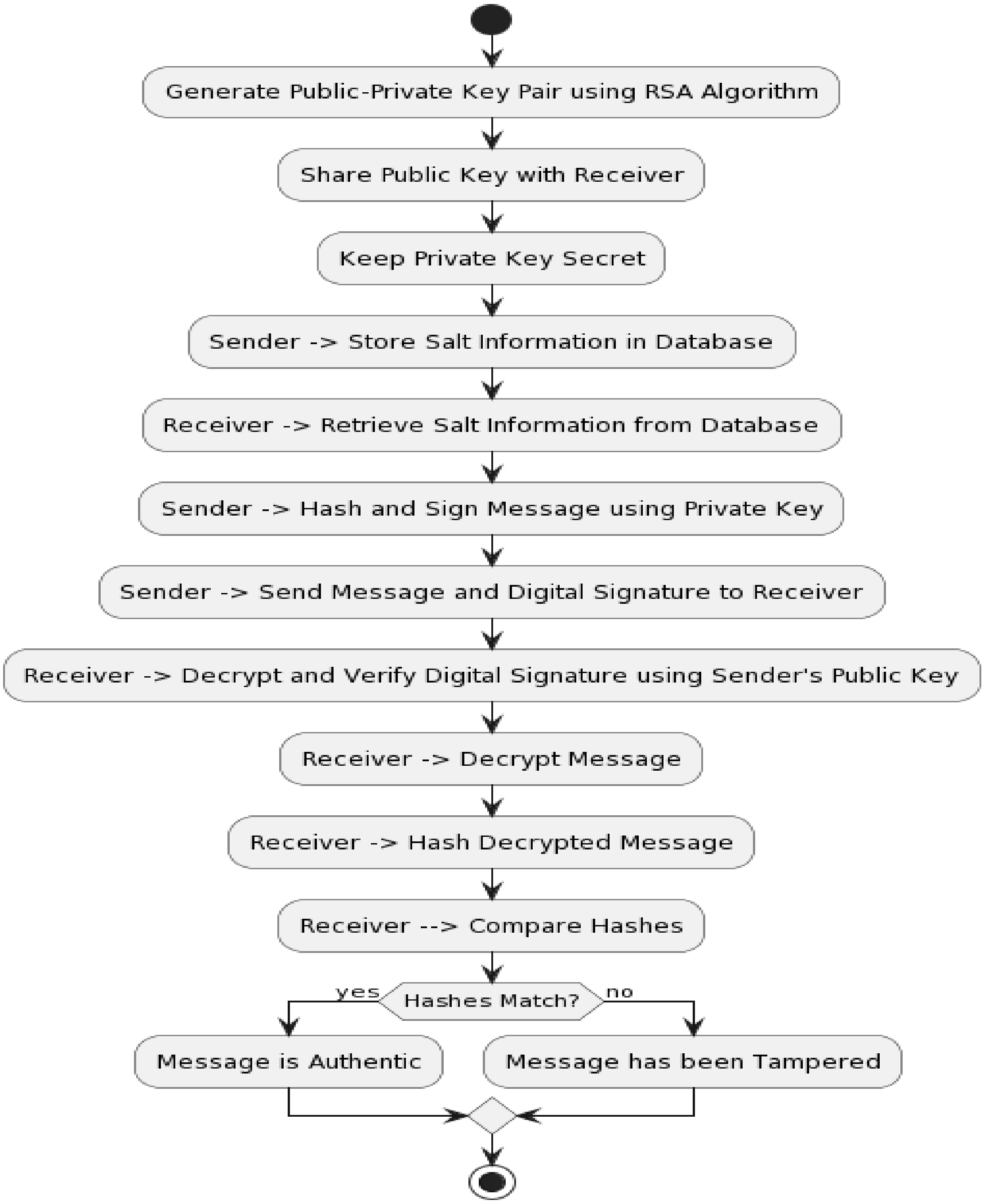

Figure 3.5: Activity Diagram

3.8 Software development model

3.8.1 Incremental Model

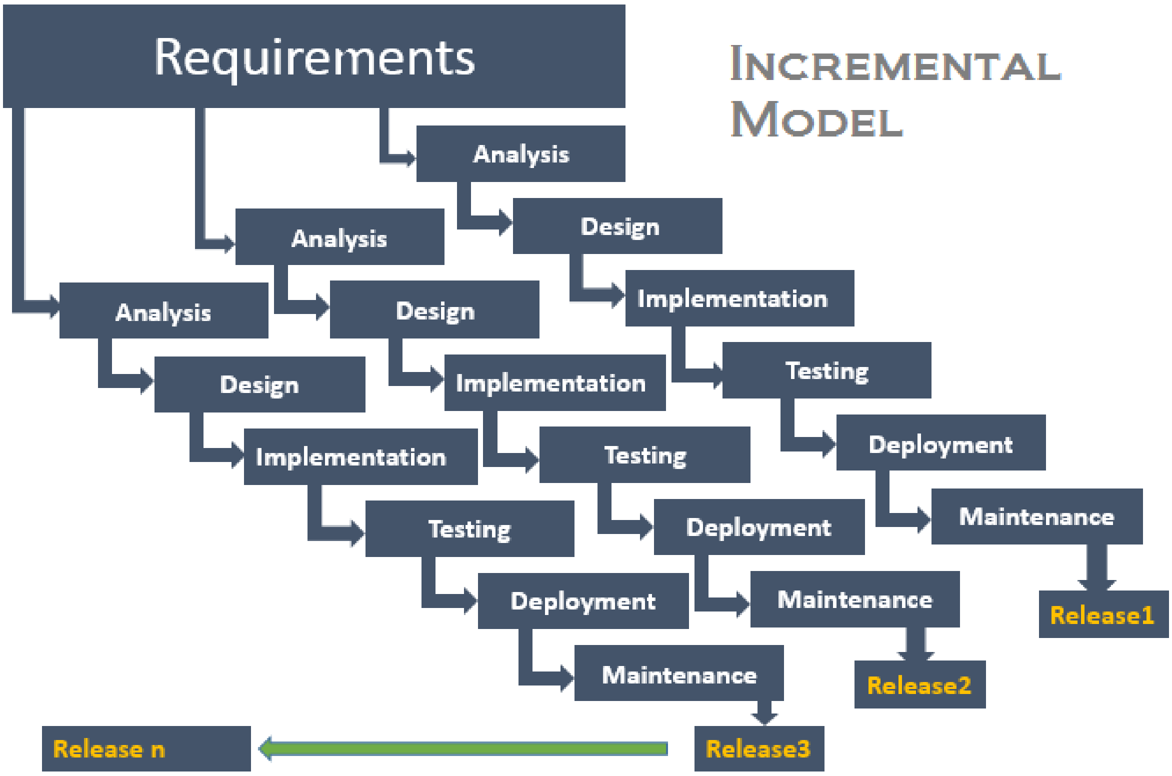

Figure 3.6: Incremental Model Block Diagram

First Increment:

First, the website was analyzed to know how it should look like. Then, the designing of templates was done. After observing the design of the templates, we started coding using react js, bootstrap, JavaScript.

Second Increment:

After analyzing the scenario of the project, the algorithms to be implemented was analyzed. Using the algorithms, we started designing the algorithms that is suitable for the project. Initiating the coding we completed the algorithm implementation. Finally, the algorithm was implemented.

Third Increment:

After the algorithm implementation backend designing and coding was started. Using Django and Python the backend part was completed and tested. Finally, backend was ready.

Fourth Increment:

Finally, after all the designing was completed, coding for the project was done. Later, testing of the project was done. At last, the final webapp was designed.

CHAPTER 4
METHODOLOGY

4.1 Required Algorithm:

4.1.1 Overview of RSA Algorithm

RSA encryption algorithm, which is based on the idea of ensuring the secure transfer of data in the digital environment and the algorithmic difficulty of separating the integer factorization, is a type of public-key encryption method. Nowadays, it is also known as both the most commonly used encryption method and the method that allows digital signatures. It was created by Ron Rivest, Adi Shamir and Leonard Adleman in 1978. Prime numbers are used for key generation process in RSA encryption method.This makes it possible to create a safer structure. How the encryption and decryption processes are done with RSA algorithm is shown in below figure,

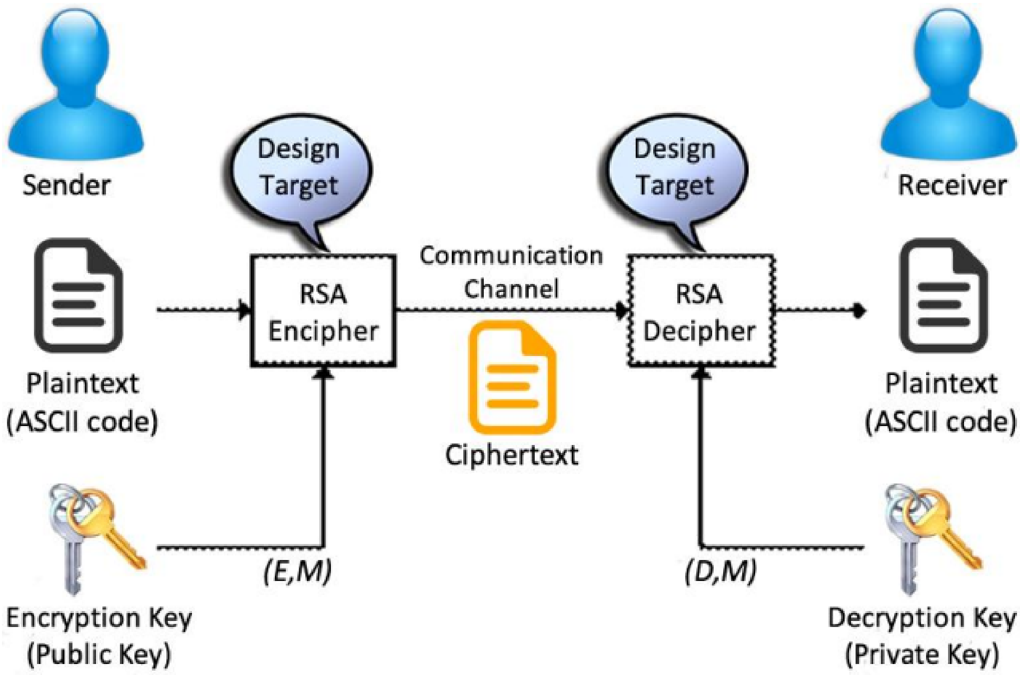

Figure 4.1: RSA algorithm working mechanism

4.1.2 RSA algorithm structure

Steps:

- Choose two very large random prime integers: p and q
- Calculate n = p*q and z = (p-1)(q-1)
- Choose a number e as public key where 1 < e < z which is co-prime with z
- Calculate d = e-1mod(p-1)(q-1)
- You can bundle private key pair as (n,d)
- You can bundle public key pair as (n,e)

After creating public and private keys, information which must be sent is encrypted with the public key.

Encryption and decryption processes are done as follows:

- The cypher text C is found by the equation where M is the original message
- The message M can be found from the cypher text C by the equation M = C^d mod n
- A text encrypted with the public key can only be solved with the private key

Salting

Salting is the process of adding random data to a password or other sensitive information before it is hashed. The salt is a random string of characters that is unique to each user and is added to the password before it is hashed. The purpose of salting is to make it much more difficult for attackers to use precomputed hash tables to reverse-engineer a password from its hash value. Without the salt, an attacker could use a precomputed hash table to compare a list of hashed passwords to a database of stolen password hashes and quickly find matches, allowing them to gain unauthorized access to user accounts. However, with a unique salt for each password, the precomputed hash tables become useless, as the attacker would need to generate a new hash table for each salt used. In summary, salting is an important security measure that helps to protect user passwords and other sensitive information by making them more resistant to attacks by malicious actors.

- Generate a public-private key pair using the RSA algorithm: The sender generates a public-private key pair using the RSA algorithm. The public key is shared with the receiver, while the private key is kept secret.
- Add Salt value in the public key: Sender adds salt value of length 6-10 to their key before sending it. The location and value of the salt is stored in the database and is retrieved during desaltification of the key
- Verify the received public key: The receiver desalts the received key using the salt information in the database and checks if the received key is correct
- Hash the message: The sender hashes the message using a hashing algorithm SHA256
- Digital Signature: The sender encrypts the hashed message using their private key. This creates a digital signature that can only he decrypted using the sender's public key.
- Send the message and the digital signature: The sender sends the encrypted message and the digital signature to the receiver.
- Verify the digital signature: The receiver decrypts the digital signature using the sender's public key. This gives them the hashed message. The receiver also decrypts the message using the same hash function used by the sender. Then the two hash function is compared and if it matches the message hasn't been tampered with.

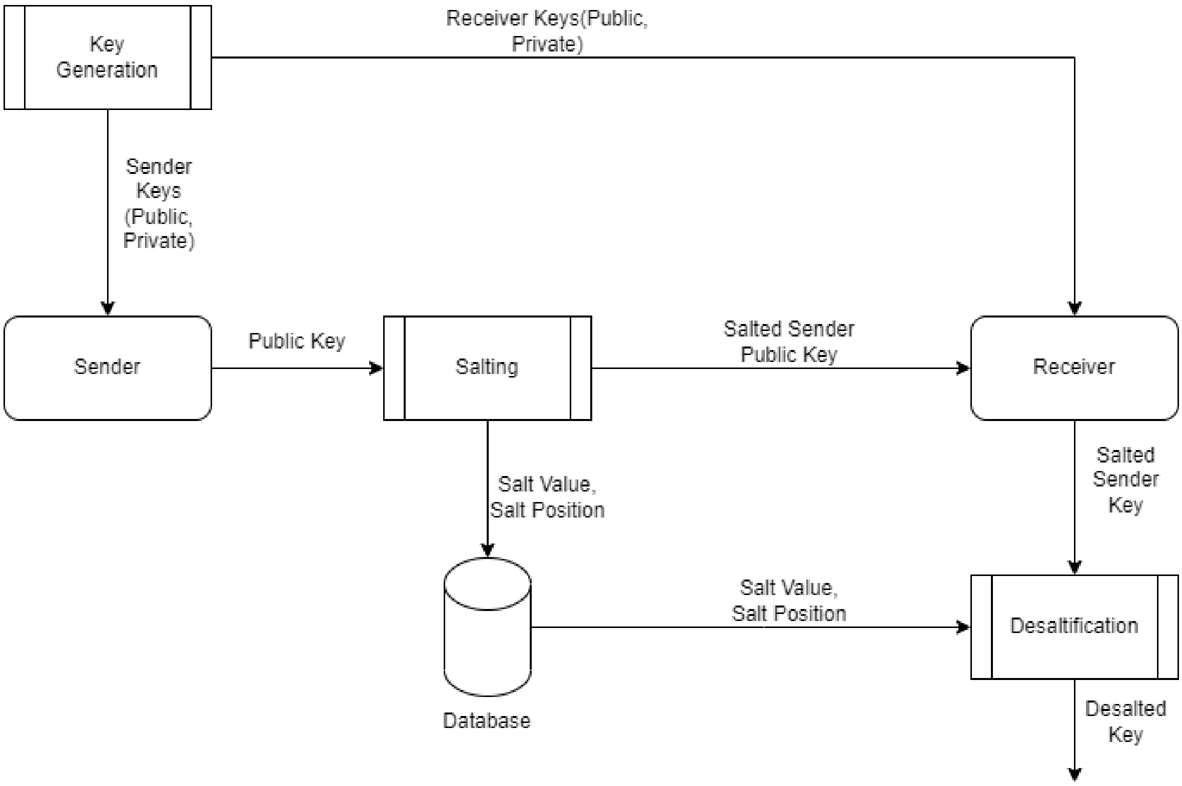

Figure 4.2: Salting Process

Hashing

Hashing is a process of converting input data of any length into a fixed-size output, called a hash or message digest. The hash function takes the input data and applies a one-way mathematical algorithm to it to produce a unique hash value. Hashing is used in many applications such as password storage, data integrity verification, and digital signatures. Hashing and RSA are two separate cryptographic concepts that are often used together in digital signatures. RSA is a public key encryption algorithm, while hashing is a one-way function used to create a unique fixed-size output (hash) from an input of any size.When we say that hashing is a one-way function, it means that it is easy to compute the hash value from the input data, but it is computationally infeasible to recover the original input data from the hash value. This property of hashing is important for ensuring the integrity and security of data. For example, in password storage, instead of storing the actual password, the password is hashed and the hash value is stored. When a user logs in with their password, the system hashes the password and compares the resulting hash value with the stored hash value. If they match, the password is correct. However, because the hash function is one-way, even if an attacker obtains the hash value, they cannot easily determine the original password. Using a hash function before applying RSA in digital signatures provides added security and efficiency, as it ensures that the signature only depends on the message's content and not on the message's size or format. Hash functions used in cryptographic mechanisms, must – depending on the application – meet the following three conditions:

- One-Way Property: For given $h\epsilon\{0,1\}^n$, it is practically impossible to find a value $m\epsilon\{\{0,1\}^*$ with $H(m) = h$.
- 2nd-Preimage-Property: For given $m\epsilon\{0,1\}^*$, it is practically impossible to find a value $m'\epsilon\{0,1\}^*\backslash\{m\}$ with $H(m) = H(m')$.
- Collision Resistance: It is practically impossible to find two values $m, m'\epsilon\{0,1\}^*$ with $m \neq m'$ and $H(m) = H(m')$.

A hash function H satisfying all of the above sconditions is called cryptographically strong. these three terms can each be described mathematically more precisely by comparing the best known attacks against these properties with optimal generic attacks. The length of the hash output is a security parameter of crucial importance, as it determines the effort of generic attacks. For the minimum security level required in this Technical Guideline of 120 bits, at least the requirement $n \geq 240$ must be imposed on hash function $H : \{0,1\}* \rightarrow \{0,1\}^n$ because of the birthday problem. At this point, it is not necessary to distinguish different cases depending on the period of use of a system, since the hash mechanisms recommended in this Technical Guideline all already have a digest length of ≥ 256 bits. [5]

Digital Signature

A digital signature is a cryptographic technique used to verify the authenticity and integrity of a digital message or document. It involves the use of a public key algorithm, such as RSA, to create a digital signature that is unique to the sender and the message being signed.

Steps:

- Creation of Hash Message: The sender of a message generates a digital signature by generating hash of the message.
- Sign Hash with private key: This hash value is signed using the priate key, resulting digital signature which is appended to the message and sent to the recipient.
- Send the message and the digital signature: The message and the digital signature are sent together to the recipient.
- Verify the signature: The recipient can verify the authenticity and integrity of the message by applying the same mathematical function to the message using the sender's public key, and comparing the resulting signature to the one that was sent with the message. If the two signatures match, it indicates that the message was sent by the sender and has not been modified in transit.

Proposed Algorithm

- Generate a public-private key pair using the RSA algorithm: The sender generates a public-private key pair using the RSA algorithm. The public key is shared with the receiver, while the private key is kept secret.
- Add Salt value in the public key: Sender adds salt value of length 6-10 to their key before sending it. The position and value of the salt is stored in the database and is retrieved during desaltification of the key
- Verify the received public key: The receiver desalts the received key using the salt information (salt value and position) in the database and checks if the received key is correct
- Hash the message: The sender hashes the message using a hashing algorithm SHA256
- Digital Signature: The sender encrypts the hashed message using their private key. This creates a digital signature that can only he decrypted using the sender's public key.
- Send the message and the digital signature: The sender sends the encrypted mes¬sage and the digital signature to the receiver.
- Verify the digital signature: The receiver decrypts the digital signature using the

sender's public key. This gives them the hashed message. The receiver also decrypts the message using the same hash function used by the sender. Then the two hash function is compared and if it matches the message hasn't been tampered with.

If the two hashes match, the message is authentic and has not been tampered with during transmission. Using RSA with salting hashing and digital signatures ensures that the message is authentic, has not been tampered with, and was sent by the claimed sender.

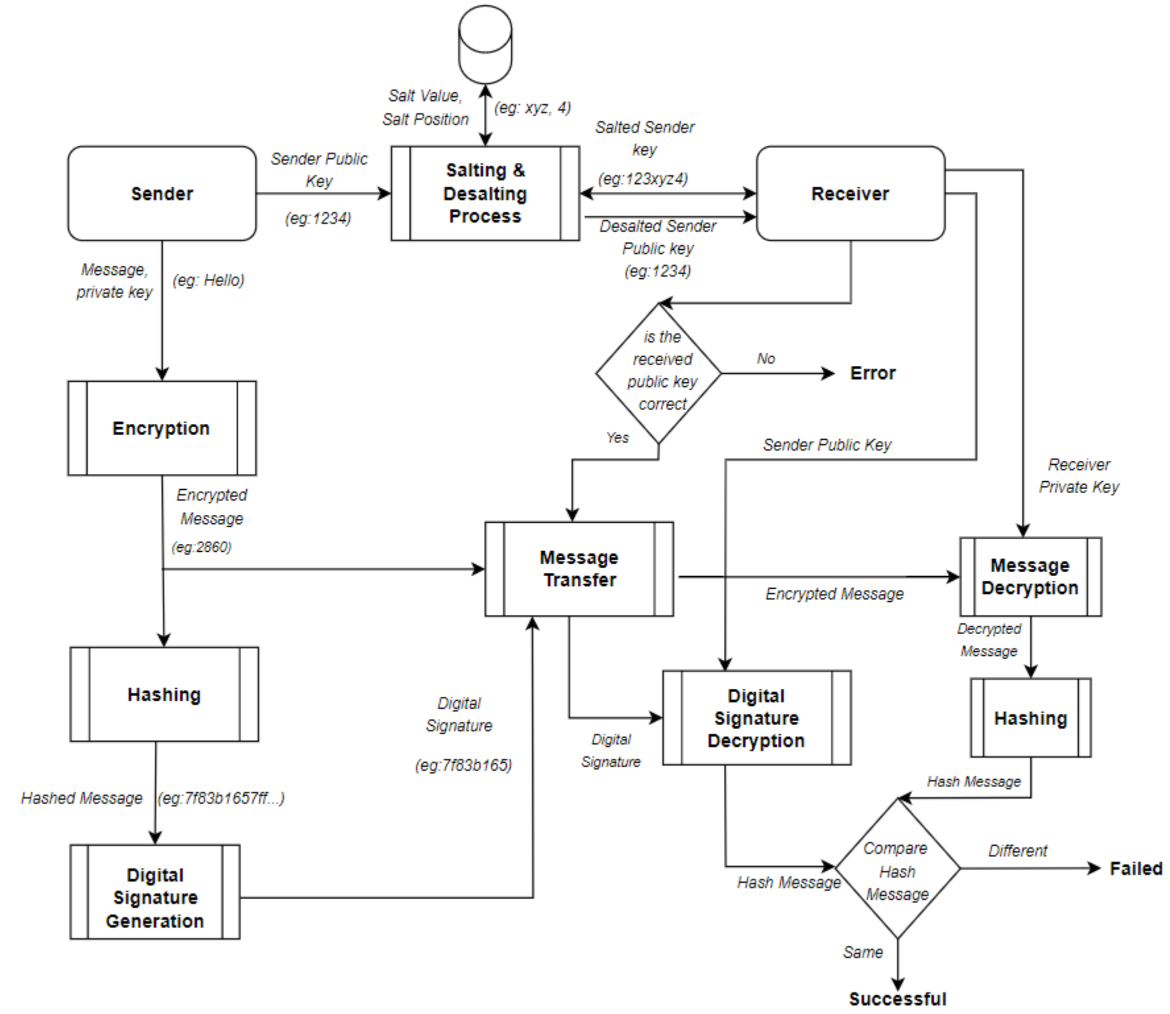

Figure 4.3: Algorithm used

4.1.3 CIA triad

CIA triad is one of the most important models which is designed to guide policies for information security within an organization.
CIA stands for : Confidentiality, Integrity, Availability.
RSA with hashing, digital signature, and salting can provide confidentiality, integrity, and authenticity (CIA) for a message or data.
Here's how each component contributes to CIA protection:

- Hashing: A secure hash function is used to produce a fixed-length hash value from the message or data. This ensures the integrity of the message, as any changes to the original message will result in a different hash value.
- Salting: Salting is the process of adding random data to the input before hashing. This makes it harder for an attacker to precompute hash values for commonly used inputs, such as passwords. Salting increases the security of the hash value, as it makes it more difficult to guess the original input.
- RSA Encryption: The hash value is encrypted using the sender's private key, which creates a digital signature. The recipient can verify the authenticity of the message by decrypting the digital signature using the sender's public key. This ensures the authenticity of the message, as only the sender's private key can create the digital signature.
- Confidentiality: RSA encryption can also provide confidentiality by encrypting the original message with the recipient's public key. This ensures that only the intended recipient can decrypt and read the message.

  Overall, RSA with hashing, digital signature, and salting can provide CIA protection for messages or data, making it a useful tool in secure communication and data protection.

CHAPTER 5
RESULT ANALYSIS AND CONCLUSION

5.1 User Application

Based on the observation of the system, we came to the following observation:

- The user application works well through all phase: signup and login, message sending and receiving. This make user experiene smooth.
- It is suitable for users who prioritize privacy in their communication. They allow users to have private conversations without fear of eavesdropping or surveillance from third parties.

5.2 Output:

In the project hour, we researched on cryptography, chat app using cryptography, their advantages disadvantages and probelms occured in chatapp etc. We developed chat app using react js, nodeJs, socket.io' and RSA algorithm and Mongodb as database where we can send end to end encrypted text message between clients.

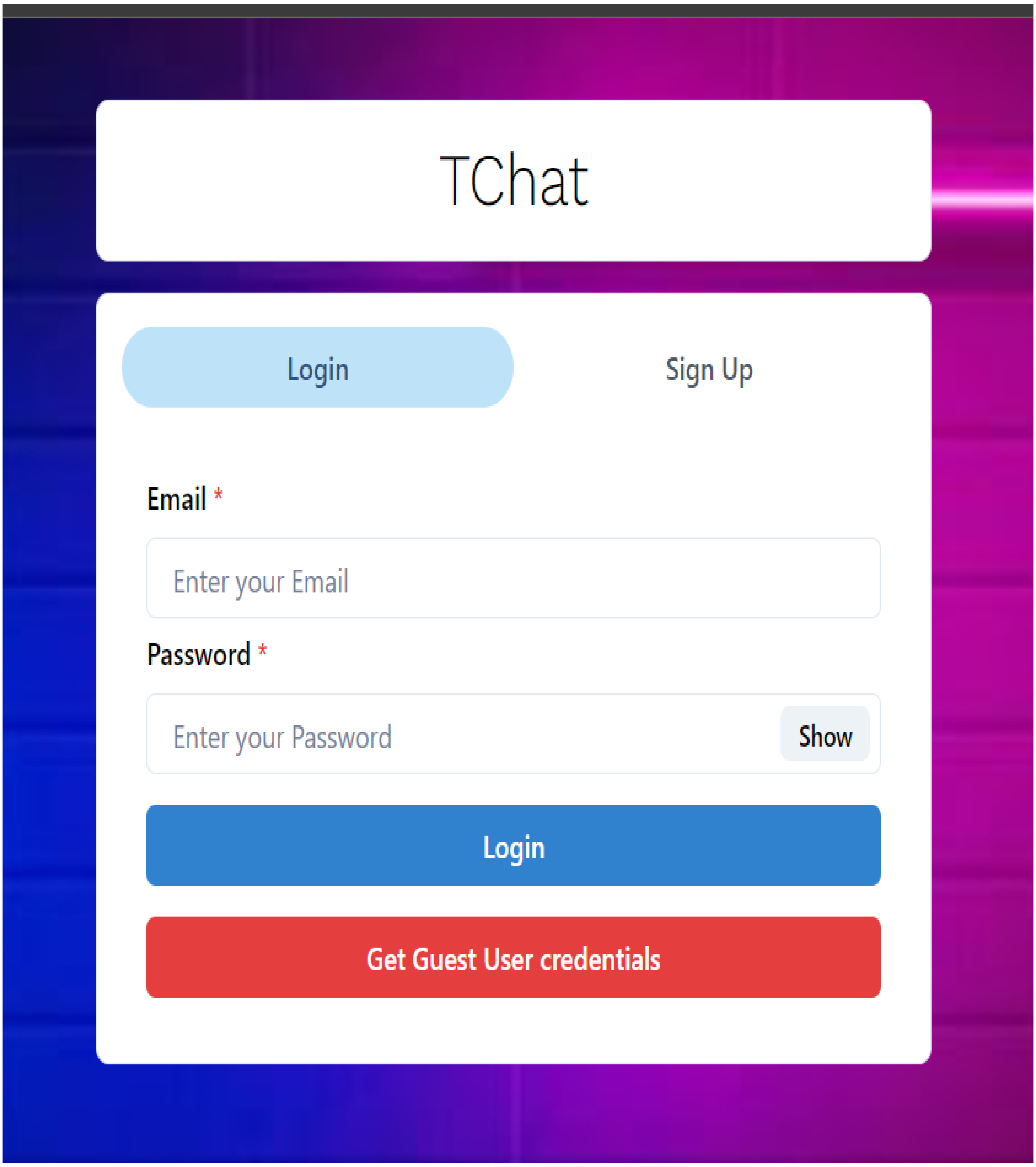

Figure 5.1: Sign in interface

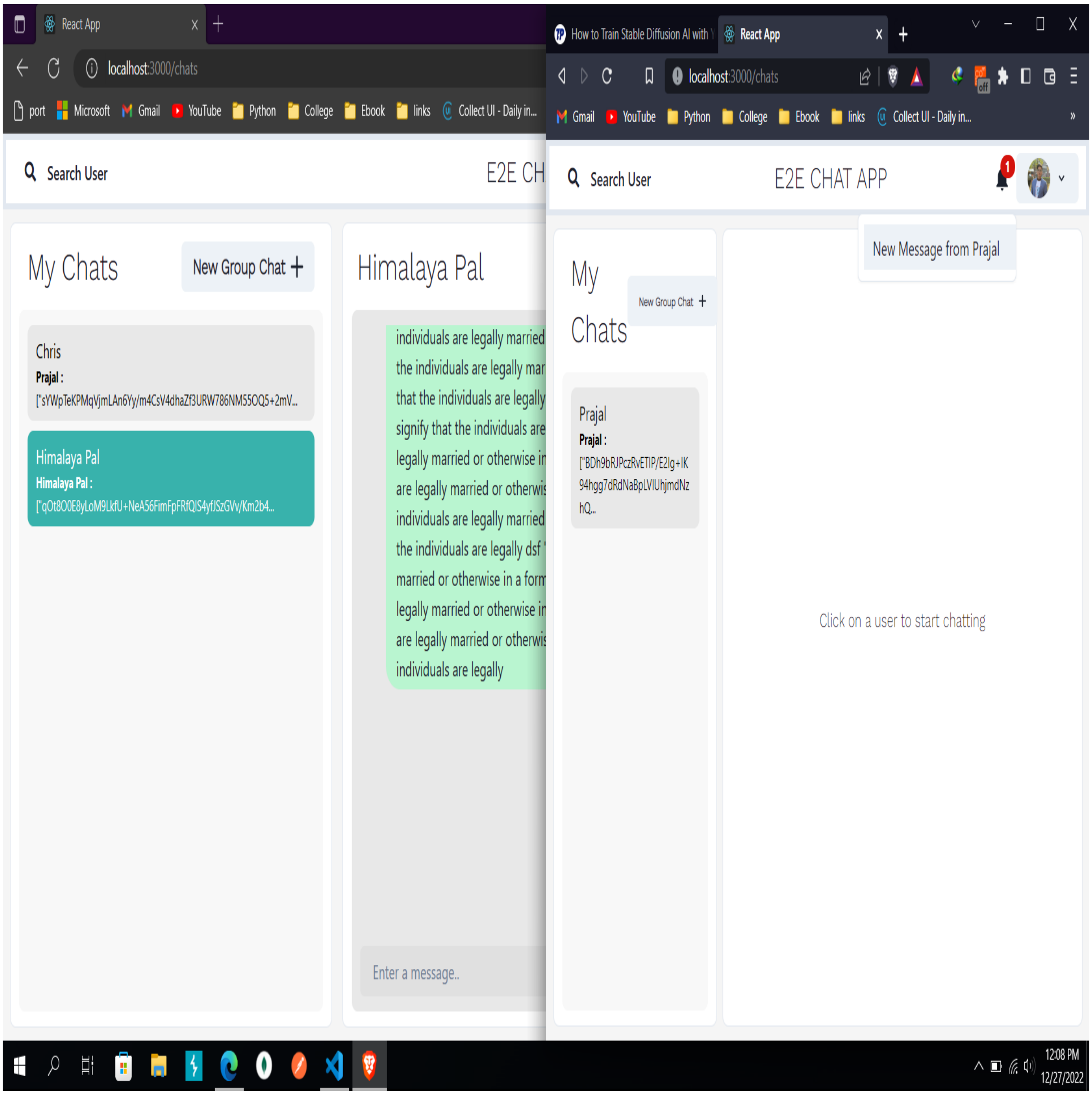

Figure 5.2: Chat interface

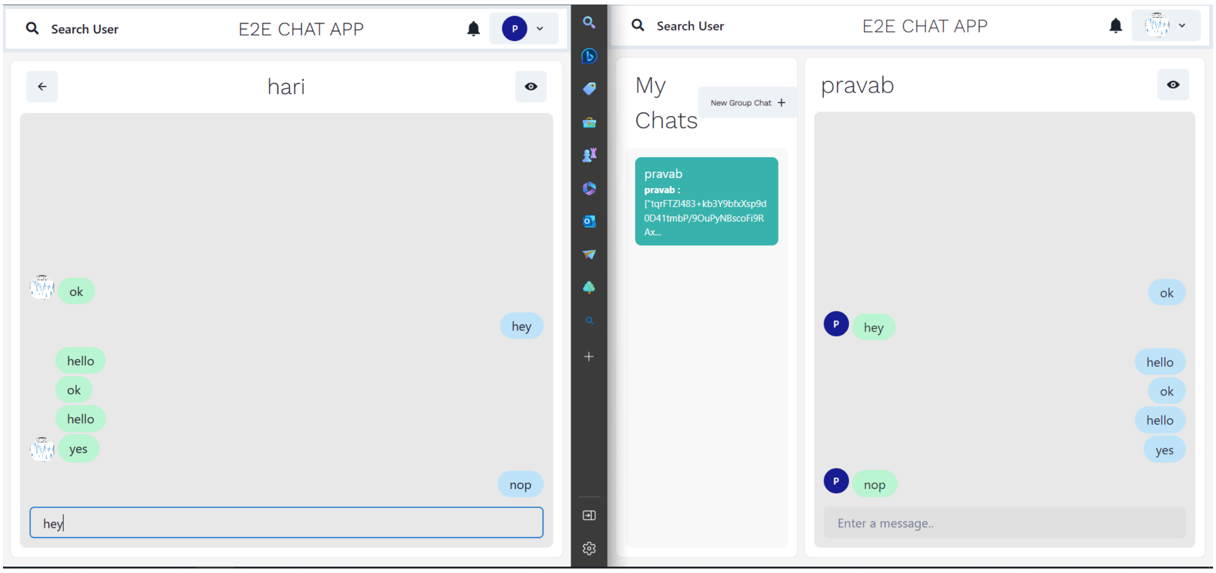

Figure 5.3: chats

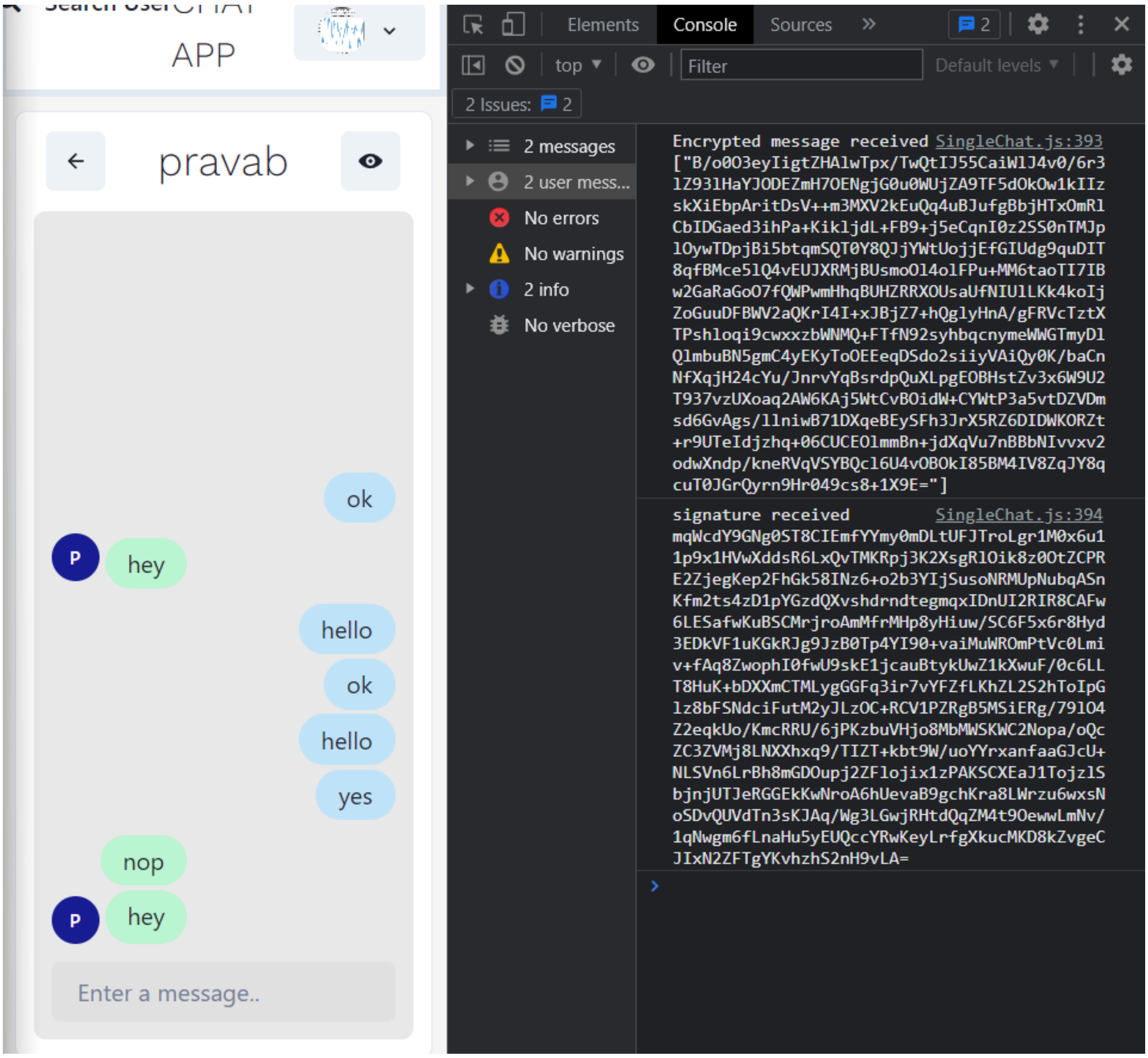

Figure 5.4: Received encrypted message and signature

5.3 Limitation

The limitation of our system are listed below:

- Our chat application can send text message with 4000 character at max. It is not available for audio, video, images messaging.
- The security measures taken can be complicated and require a lot of computing resources to implement, which can make the chat application bit slower to use.
- The system required the first internet connectivity for smooth operation.

5.4 Future Enhancement

This project can be further enhanced. Currently, we are able to send text message only but it can be developed to send audio, video , images messaging chat app. We can add group chats options. As well as this project can be made more secure with updates. Extra new trending features can be added in our system. We can also made this application compatible to all kind of platforms and devices.

5.5 Work Schedule

Scheduling establishes the timelines, delivery and availability of project resources whether they be personnel, inventory or capital. For this reason, any project without a schedule is a project doomed to issue down the road. The estimated time period of this project is 30 weeks. The work is divided into several phases as shown in gantt chart.

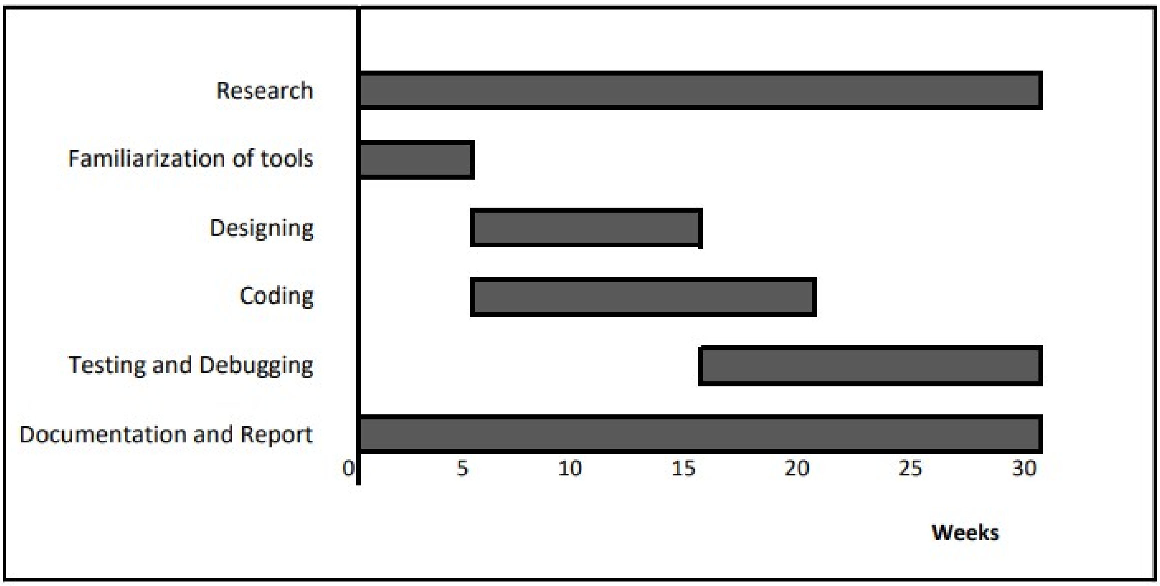

Figure 5.5: Gantt Chart

5.6 Conclusion

With the increasing importance of privacy and security in the digital age, it is imperative that we offer users a secure means of communication. Our proposal offers a reliable and effective solution that can help individuals and organizations protect sensitive information and communicate with confidence. We believe that our proposed end-to-end chat application will be a valuable addition to the landscape of secure communication tools, and we look forward to further development and implementation of this technology.

REFERENCES

[1] O. O. Blaise, O. Awodele, and O. Yewande, “An understanding and perspectives of end-to-end encryption,” *Int. Res. J. Eng. Technol.(IRJET)*, vol. 8, no. 04, p. 1086, 2021.

[2] N. Sabah, J. Mohamad, J. M. Kadhim, and B. N. Dhannoon, “Developing an end-to-end secure chat application,” 2017.

[3] F. Sönmez and M. K. Abbas, “Development of a client/server cryptography-based secure messaging system using rsa algorithm,” *J. Manag. Eng. Inf. Technol*, vol. 4, no. 6, 2017.

[4] E. Barker and Q. Dang, “Nist special publication 800-57 part 1, revision 4,” *NIST, Tech. Rep*, vol. 16, 2016.

[5] C. M. Bsi, “Cryptographic mechanisms: Recommendations and key lengths,” *BSI—Technical Guideline*, 2020.