KATHFORD INTERNATIONAL COLLEGE OF

ENGINEERING AND MANAGEMENT

Balkumari, Lalitpur

A

Minor Project Proposal

On

“Realtime Chat Application with end-to-end encryption”

Project Members

Satish Adhikari (075/BCT019)

Manoj Chhetri (075/BCT014)

Shree Ranjan Labh (075/BCT022)

DEPARTMENT OF COMPUTER AND ELECTRONICS &

COMMUNICATION ENGINEERING

LALITPUR, NEPAL

JUNE 2021

Abstract

"Real Chat" is a web application which facilitates asymmetric encryption for messaging. This app can be used in private networks for messaging in insecure channels and public Wi-Fis, and in the Internet through a web server. The application is designed to enhance the user experience and ensure that users get real-time messaging service as secure as SSH. Like the RSA algorithm, we have used Diffie Hellman's Algorithm to facilitate end-to-end encryption through the use of private and public keys for encryption and decryption of messages. The app lets the user to register and login through their input credentials (username and password). After logging in, users are provided with a clean and simple UI which lets them message in real-time using web sockets. Although there are already heavily used messaging apps in the real world, this app still facilitates hard rock encryption as strong as WhatsApp's and a means for messaging in private networks. Without storing messages unless for attached files, we intend to opensource our app and provide a transparent exhibition of secured messaging. Furthermore, a contacts-keeper utility will be facilitated for easier communication with amiable users.

Keywords: *Asymmetric encryption, web server, SSH, RSA algorithm, end-to-end encryption, UI, private networks, opensource.*

Table of Contents

List of Figures

List of Abbreviations

API	Application Programming Interface
E2EE	End-to-end Encryption
IDE	Integrated Development Environment
URL	Universal Resource Locator
WWW	World Wide Web
SSH	Secure Shell
RSA	Rivest-Shamir-Adleman
UI	User Interface

1 Introduction

1.1 Background

The Real Chat Application is intentionally proposed to establish a secure means of messaging within private networks. Since there are already numerous messaging services available out there in the Internet, none of them can really be trusted for our private data. Although this web server can also be hosted in the Internet and used like other messaging services, we highly recommend its usage within private networks where messages aren't by default encrypted.

Introduction to Diffie-Hellman Algorithm:

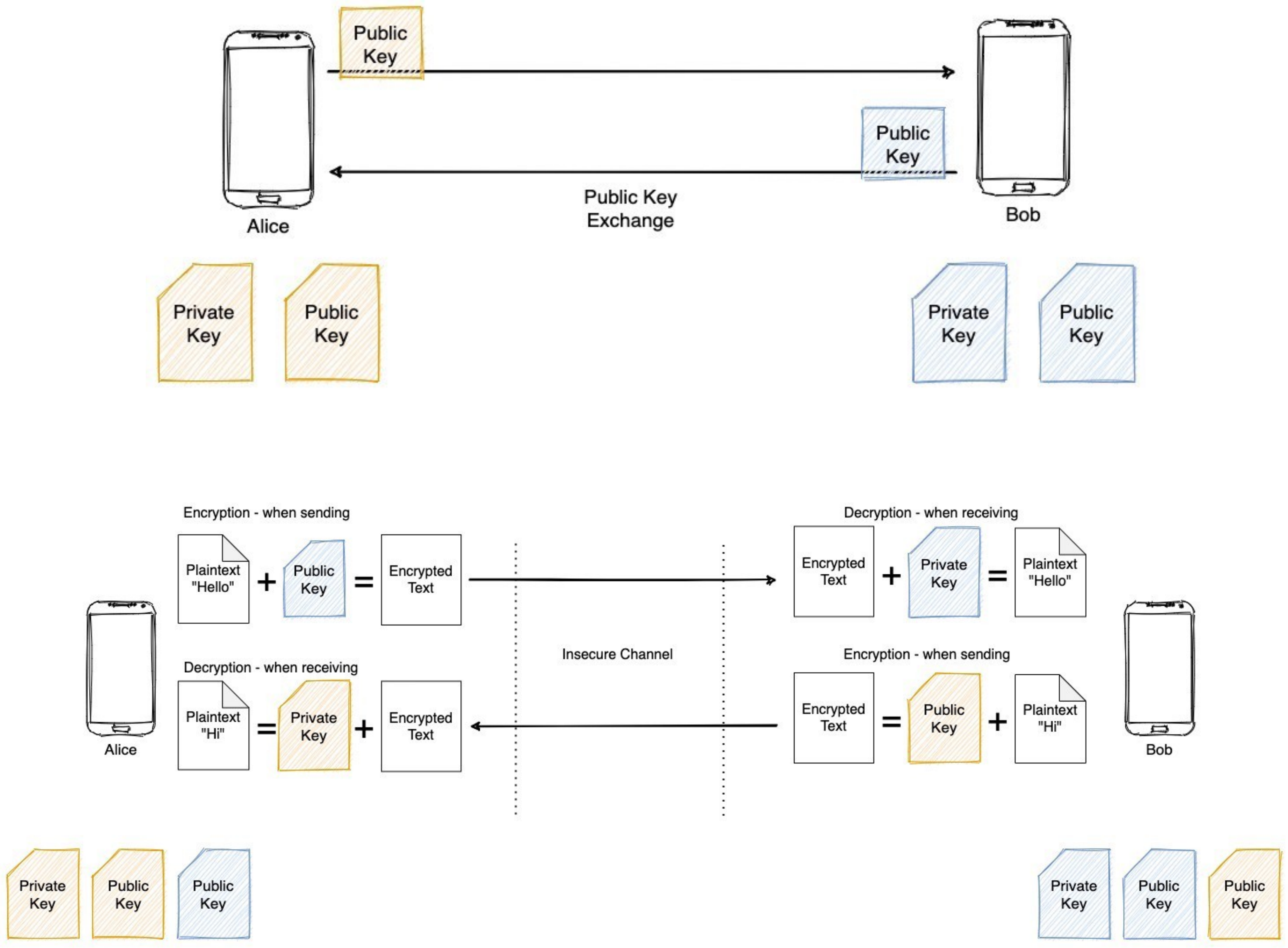


Figure 1.1 Working Diagram of the Asymmetric Encryption (I)

Asymmetric encryption requires two types of keys — a **public** and a **private** key-pair. The following diagrams explain the process of

1. Public key exchange
2. Sending and receiving messages using this method.

Messages in E2EE can be **encrypted using the receiver's public key** at the sender's end so that it can only be **decrypted at the receiver's end using his/her private key**.

The public keys of users can be stored in the database and can be serviced upon client's request while the private keys can reside on users' devices.

But how will Alice and Bob share the common key via the insecure channel? This where ***Diffie-Hellman key exchange method*** *comes in*. The objective is to obtain the common key without even exchanging the key at the first place.

This is done by generating the key at the clients' side independently and ending with a same key! It involves:

- A large **prime number (P)**,
- It's **primitive root (g)** and
- A **private key** for each client.

(***P)*** and (***g)*** can be exchanged via the insecure channel.

Alice generates a number **(A)** at her end using her private key **(a)** as `A = g^a modulo P`, where ^ denotes `power of`. Likewise, Bob generates a number **(B)** at his end using his private key **(b)** as `B = g^b modulo P`.

Now they can share this newly generated keys (***A)*** and (***B)*** via the insecure channel as no one can reverse engineer it to get back their private keys **(a)** or **(b)**.

This is so because even if you know the values of **(g)** and **(P)**, it becomes **practically impossible** to deduce **(a)** or **(b)** since you don't know how many revolutions `g^a modulo P` or `g^b modulo P` made between `0` and `P-1`.

At this point, **Alice receives (B)** and **Bob receives (A)**. Alice can now generate the common key by `common key = B^a modulo P` at her end. Likewise, Bob can generate the same using `common key = A^b modulo P` at his end. **The two formulae will generate the same result!** Now both of them can exchange messages by encrypting and decrypting using this newly generated common key.

Using this asymmetric encryption, **web sockets [1]** will be used alongside to establish real-time messaging between users in the backend and a **User Interface** will be used to facilitate users for easy communication.

1.2 Problem Statement

Within a private network, it’s very common to send unencrypted plain text messages that can be instantly sniffed and tampered with, firing some tools like Wireshark, tcpdump. This can even enable attackers to conduct a ‘Man in The Middle’ attack in between the two communicators. Wide in the Internet, several encryptions take place to safely send our messages over to other users. However, we can never rely on any such service to not misuse our personal data, given many such cases!

Given recent data breaches and the importance of data privacy, it’s been tough to trust data communication and online services. Any private data with an access to the Internet, should be considered unsafe.

1.3 Project Objectives

- To design and implement a real-time chat application using encryption to send messages a within a network for many users.

2 Literature Review

Many of us could believe that all the confidential data shared is safe via Facebook Messenger, Skype or Snapchat, but sometimes it's just an illusion. The recent events in which Facebook shared users' private information with Cambridge Analytica in what seems to be one of the social network's largest data breaches, should make us more aware of the importance of data privacy. **Without end-to-end encryption**, our conversations most likely will get into the hands of cybercriminals, and other malicious actors focused on stealing them.

Martin Kleppmann, former Rapportive co-founder, and LinkedIn engineer sums up the value of end-to-end encryption in a **great blog post [2].** His comparison between different types of data encryption is also useful to explain the difference between the widely used **encryption in transit** and the more secure **end-to-end encryption** process.

Diffie–Hellman key exchange [3] is a technique used to achieve asymmetric encryption that is used to produce the most secure chat applications of all times including WhatsApp and Telegram. It is as secure as the RSA algorithm, which also involves private and public keys for secure communication and is used for secure apps like Secure Shell (SSH). **This article [4]** best explains Diffie-Hellman algorithm in chat applications using **Web sockets** in real time.

3 Feasibility Study

3.1 Technical Feasibility

Evaluating the technical feasibility at this early stage is the trickiest part of feasibility study. This application is technically feasible. The services will be provided through the application. The services that are intended to be provided are possible to be implemented using the available technology.

3.2 Economic Feasibility

This application is economically feasible since we will be using free software and publicly available data for the project. So, this project can be accomplished at no cost. The startup capital needed is none.

3.3 Schedule Feasibility

Schedule feasibility is a measure how reasonable the project timetable is. This project is completely schedule feasible too as an app can be made under a year so, under a year this app will be completed.

3.4 Operational Feasibility

This application satisfies the users requirement and can be fitted into the current system operation that can enhance user capability. The system is user friendly so the user can use it more effectively.

4 Project Methodology

4.1 How It Works!

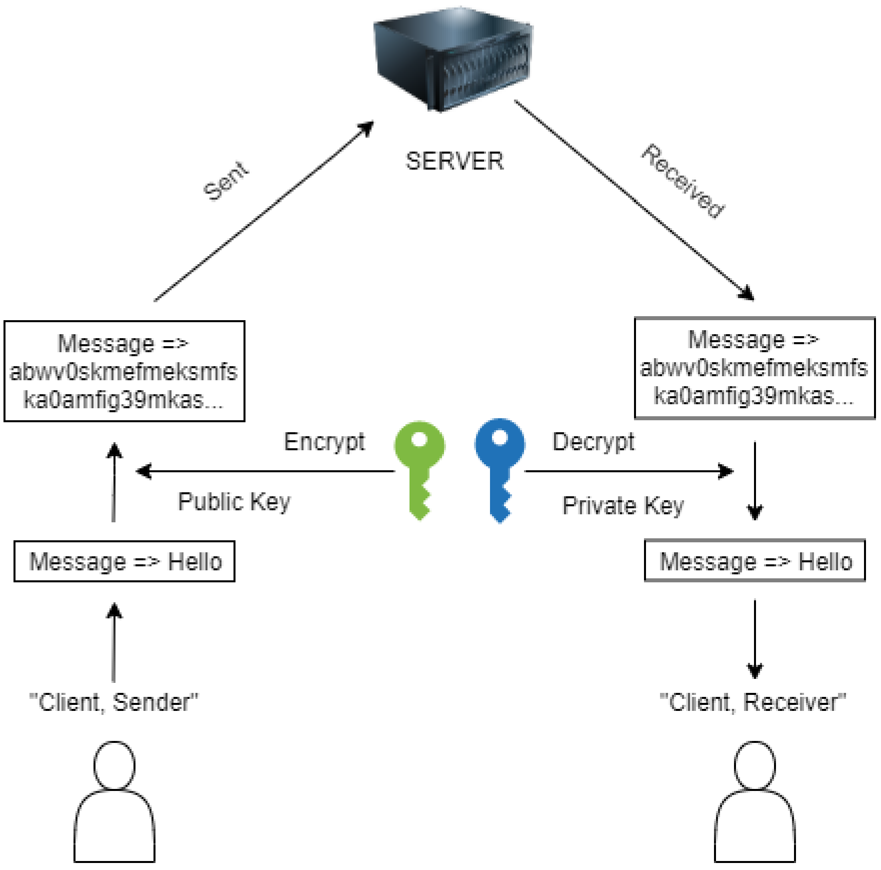
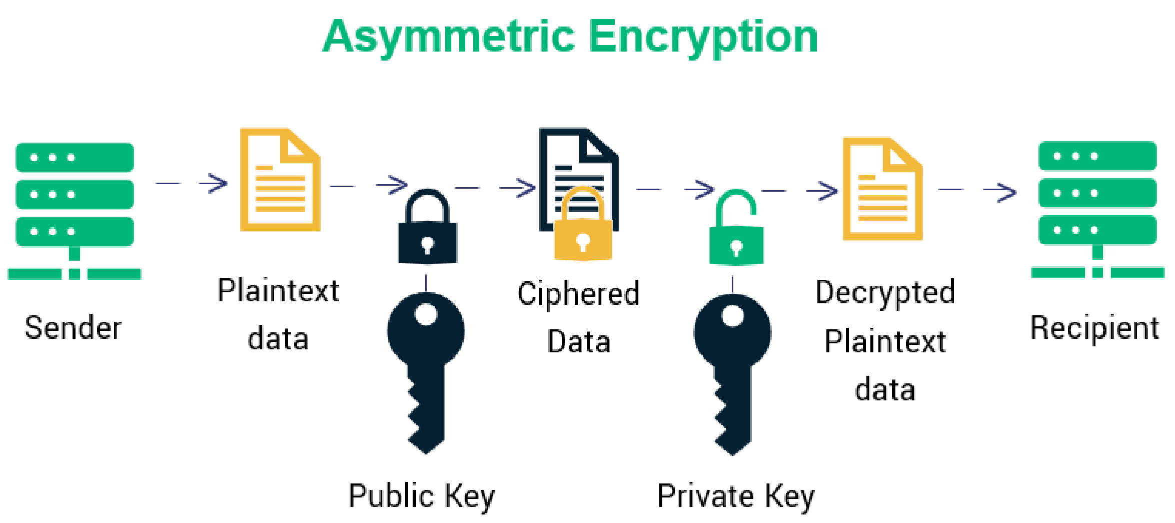


Figure 4.1 Working Diagram of the Real Chat Application

As in the diagram, two clients (sender and receiver) can send messages to each other. Two keys, public and private are used for encryption and decryption respectively. They function to encrypt and decrypt the message in both client's application and thus the target message is transferred without getting tampered with!

For a better UX (User Experience), simple and easy to use UI has been decided which only includes the functions that the users require. For a proper security an authentication system will be added. Web sockets will be used in the backend of the application to facilitate real-time messaging and communication. Users will log in and an end-to-end encryption with Diffie Hellman's algorithm will be established between the sender and receiver for messaging using private keys and sharing public keys for decryption and encryption purposes. The figure 4.1 below shows a asymmetric encryption between the clients in action.



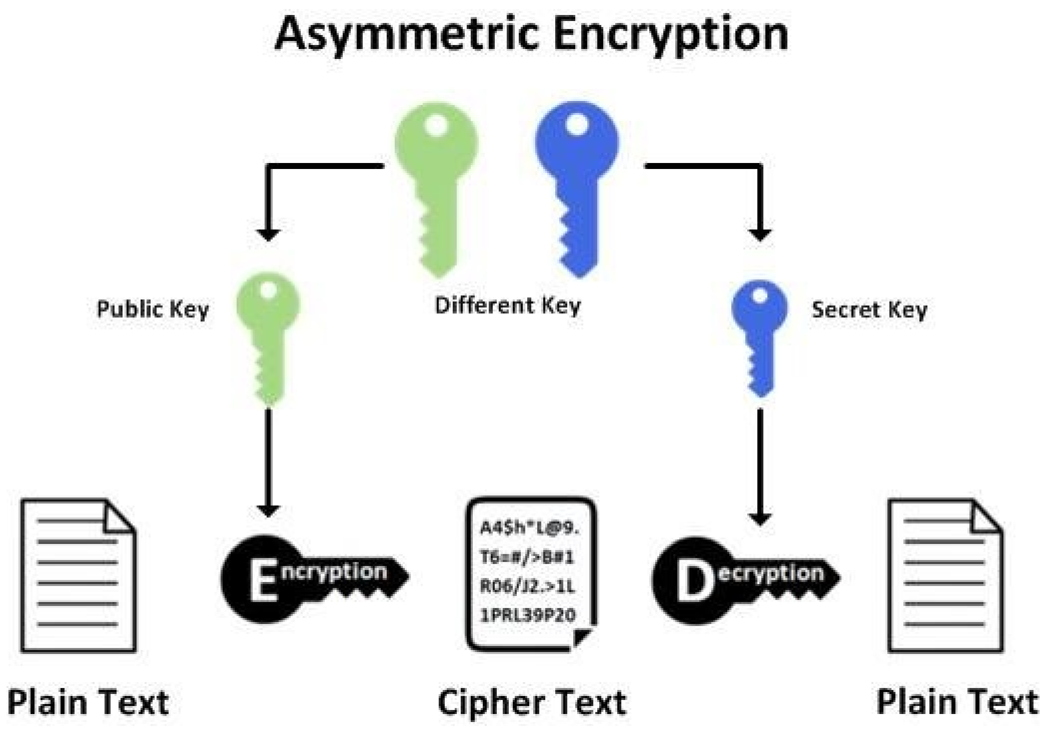


Figure 4.2 Working Diagram of the Asymmetric Encryption (II) [5]

The strategy for implementing stated encryption and algorithm is straight forward. First, an application that has a user-friendly interface will be created. Users can be created and logged in to see their contacts list and communicate securely even in an insecure channel. This application maybe hosted in WWW for spanning across networks or basically be used as the primary way to securely conduct messaging.

4.2 Development Model

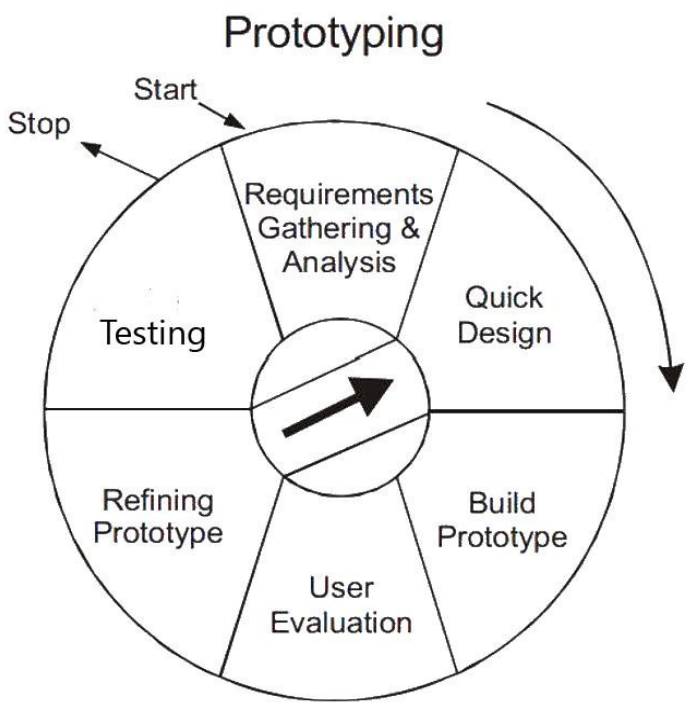


Figure 4.3 Prototype Model of Development

Prototype Model of software development provides flexibility in the development process by allowing the user to interact and experiment with a working representation of product known as prototype. As the technology that will be utilized in this project is new to us, we will be unable to get the best outcomes in our first endeavor. We should keep tweaking with the parameters of the model until we receive satisfactory accuracy, that is the reason we have picked the Prototype Model as the Software Development Life Cycle (SDLC). This will enable us to better define our user experience and their contacts, which can always be improved even better.

5 Implementation Plan

5.1 Requirements

The whole project will be implemented in JavaScript programming language, using its React Framework in the Frontend, while NodeJS in the Backend.

Implementation Tools:

I. Frontend tools

- ReactJS, Redux
- Figma

II. Backend tools

- NodeJS
- Redis, MongoDB if necessary.

III. Other tools

- Microsoft Word 2019
- Microsoft Visio 2019

6 Expected Outcomes

The outcomes that can be expected by the user of this application are:

- The application users will be able to communicate with end-to-end encryption to send messages and files within networks.

- Users will be able to allow other users to be their contact and implement a contacts-keeper utility.

References

[1] "Wikipedia - Web sockets" [Online]. Available:
https://en.wikipedia.org/wiki/WebSocket

"Mozilla Developers - Web sockets API ":
https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

[2] “User Interface” [Online]. Available:
https://www.interaction-design.org/literature/topics/ui-design

[3] “Martin Kleppmann – End to End Encrypted Communication" [Online]. Available:
https://martin.kleppmann.com/2015/11/10/investigatory-powers-bill.html

[4] "Wikipedia – Diffie Hellman Key exchange" [Online]. Available:
https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

[5] "Medium – Diffie Hellman Key Exchange in E2EE” [Online].
Available: https://shubhomoybiswas.medium.com/diffie-hellman-key-exchange-in-end-to-end-encryption-e2ee-2366e056661