Contents

[1 Analysis 2](#_Toc64045661)

[1.1 Problem Identification 2](#_Toc64045662)

[1.1.1 Problem outline 2](#_Toc64045663)

[1.1.2 Stakeholders 2](#_Toc64045664)

[1.1.3 How can the problem be solved by computational methods? 2](#_Toc64045665)

[1.1.4 Computational methods that the solution lends itself to 2](#_Toc64045666)

[1.2 Research 4](#_Toc64045667)

[1.2.1 Existing program – Discord 4](#_Toc64045668)

[1.2.2 Existing solution – Internet Relay Chat (IRC) 5](#_Toc64045669)

[1.2.3 Interview questions 5](#_Toc64045670)

[1.2.4 Interview 5](#_Toc64045671)

[1.3 Requirements 5](#_Toc64045672)

[1.3.1 Software requirements 5](#_Toc64045673)

[1.3.2 Stakeholder requirements 5](#_Toc64045674)

[1.4 Success Criteria 5](#_Toc64045675)

[2 Design 5](#_Toc64045676)

[3 Development 5](#_Toc64045677)

[4 Evaluation 5](#_Toc64045678)

[5 Bibliography 6](#_Toc64045679)

# Analysis

## Problem Identification

### Problem outline

Currently, the encryption in-transit system is widely used in instant messaging apps. This means that a user’s messages will be in plaintext in the service provider’s server, which allows the service provider to read users’ conversations breaching their privacy. These companies are then also susceptible to potential attackers accessing confidential information through this potential backdoor. Therefore, I will be making a solution to this problem of user privacy by making an end-to-end encrypted instant messaging program.

For this solution to work, the following features are required: only the endpoint users have the key for the symmetric encryption algorithm to qualify the system as end-to-end encryption; an easy-to-use GUI so that anyone can navigate and use the program; a login/sign-up system to authenticate users’ identities to the server to facilitate historic message viewing; networking capabilities in order send and receive messages from different networks with a low enough latency to be a viable ‘instant’ messaging platform.

### Stakeholders

The target audience of this program is split into two categories. Firstly, there will be privacy-conscious stakeholders who are choosing their instant messaging platform based on the security that the program provides to their information. The requirements for these stakeholders will be the following: all the communications on the program will be encrypted; the only parties that can view the plaintext messages are the endpoint users; all sensitive passwords stored on the server will be hashed. These stakeholders will make use of the program through business, gaming, or everyday communications to name a few use cases as all the needs are satisfied as part of the program’s essential features.

Secondly, there will be stakeholders who are people less familiar with computers that are looking for an easily accessible instant messaging platform. The requirements for these users will be an intuitive user interface with little to no prior setup needed for them to start using the program. They will make use of the program to keep in touch will family and friends; therefore, the program must be easy to use as to not deter them.

### How can the problem be solved by computational methods?

This problem is well suited to a computational approach as it can be solved using computational methods. This is because the process of encrypting, sending then decrypting messages over a network can be greatly abstracted for the user so that they do not need any prior technical knowledge to use the program. The solution will also have many algorithms to carry out processes without the need for complex input from the user.

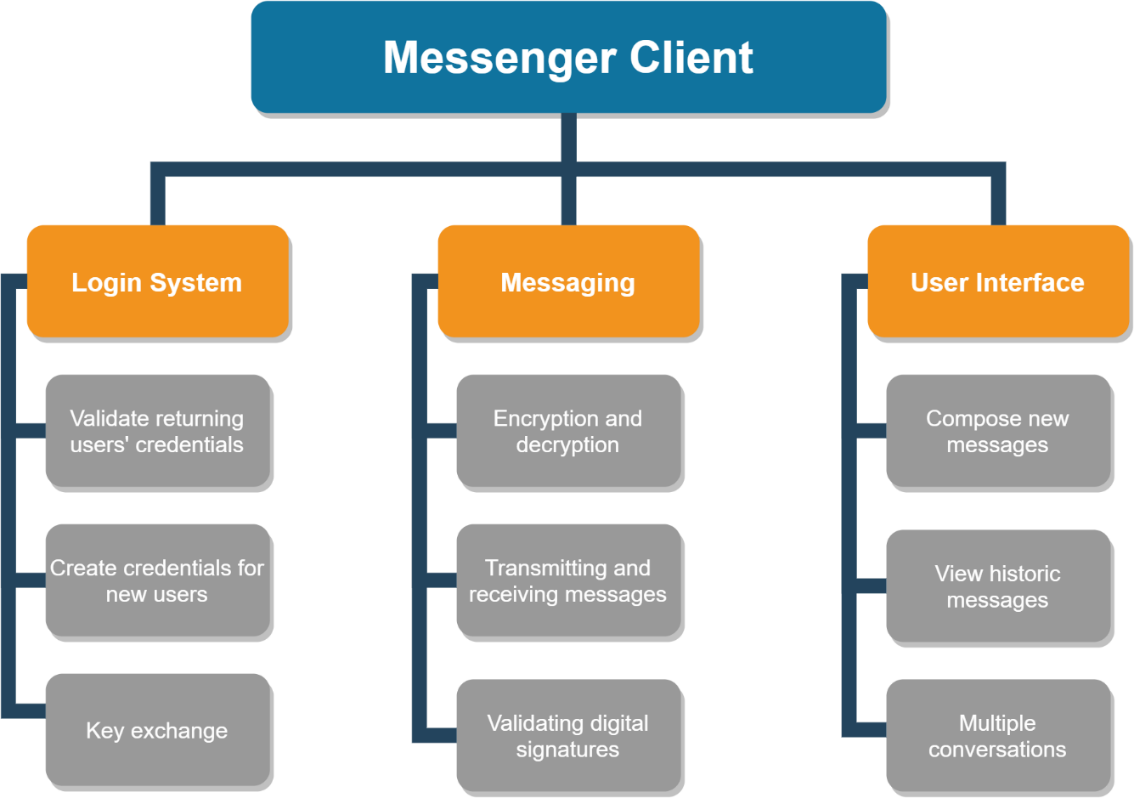
### Computational methods that the solution lends itself to

**Problem recognition** – The general problem is creating a program where sensitive information is not ever in plaintext whilst passing through the server. However, the more specific problems are creating a system of generating two symmetric private keys over a network connection that people monitoring the network are unable to recreate and sending and receiving the messages over that network. Once this is overcome the rest of the solution is using those keys to do the encryption/decryption and presenting the messages to the user in an intuitive form.

**Abstraction** – It is not necessary (or feasible) for the user to have hands-on control over every single process that happens when sending their messages. Therefore, abstraction will be used to hide most of the processes from the user to streamline their experience by only showing relevant detail. Some of these processes will be the following: encrypting and decrypting their message; segmenting their messages into packets; sending the packets to the server.

**Thinking ahead** – I have thought ahead by choosing C# to write the program is since I expect the backend to require a lot more focus than the frontend; the GUI capabilities in C# are much greater than Python for example (the language I am most familiar with), due to its support for WYSIWYG GUI builders in Visual Studio. I made this decision so that I can spend more time focusing on the development of the backend.

**Procedural thinking & decomposition** – The problem can be decomposed into a set of much smaller problems, using procedural thinking I will develop a solution to each of these in turn.



This set of small problems will allow me to possibly use test harnesses during development to isolate certain parts of the program during the development process. This structure will make the overall solution easier to work with and will make the entire process more efficient.

**Thinking concurrently** – Through the use of concurrent processing, I will make the server-side program more efficient by processing each user’s requests on a virtual thread. This will mean that the program can deal with requests from multiple users and the same time, which is important as if they were acted on procedurally the bandwidth of the system would be dramatically reduced leading to very high latency.

**Performance modelling** – I will make heavy use of performance modelling to ensure that the program functions efficiently. On the smaller scale, I will use performance modelling to profile my encryption algorithms to make sure that they properly balance processing time and security since for this application an algorithm that sufficiently encrypted data in a few milliseconds is vastly superior to an algorithm that more complexly encrypts data in a few minutes. On a larger scale, I will use performance modelling to ensure that my server-side program has properly utilised threads to have a suitable bandwidth to support both average and peak throughput.

## Research

### Existing program – Discord

**Overview:**

Discord is a free instant messaging and VoIP platform created in 2015 centred around enabling communities to connect through guilds: collections of chat rooms and voice channels. In 2019 the platform saw 250 million users with a total of 25 billion messages being sent per month [1] making it one of the largest gaming-focused communications platforms available.



**Features:**

Each guild is managed by users with varying levels of permissions which can be as general or customised as desired by the guild owner. In the text channels, you can send files, links, and text with basic mark-up. In voice channels, as well as streaming audio you can also use webcams and stream your desktop. The purpose and scope of a guild very flexible as there is no predefined structure: they can be created for a small group of friends or as large official game hubs.

**Differences:**

Discord is not a privacy-focused platform and users are expected to forfeit their privacy in exchange for ease of use and versatility. Discord uses the encryption in-transit system meaning that all traffic is decrypted on the server-side; for non-audio/video data, the HTTPS protocol is used which is encrypted using TLS or SSL. It is also known that Discord inspects all user traffic whilst it passes through their server. However, the reason why is unknown.

**Parts I can apply to my solution:**

1. The idea of having a central area consisting of multiple ‘channels’ for different conversations or topics would potentially be worth including in my solution. This is because it would be an intuitive structure for larger chat rooms, where having a singular area for everyone to talk in would not be feasible.
2. I also think that Discord’s feature of activity statuses for each user would be a good addition to my solution. This would not be an essential feature but, if I have enough time to add it, it will improve the overall experience of the user by enabling them to see who is also online while they are using the program.

### Existing solution – Internet Relay Chat (IRC)

### Interview questions

### Interview

## Requirements

### Software requirements

### Stakeholder requirements

## Success Criteria

# Design

# Development

# Evaluation

# Bibliography

|  |  |
| --- | --- |
| [1] | C. Corberly, “Discord has surpassed 250 million registered users,” TechSpot, 13 May 2019. [Online]. Available: https://www.techspot.com/news/80064-discord-has-surpassed-250-million-registered-users.html. [Accessed 12 February 2021]. |