

Wentworth Institute of Technology
COMP4960 – Software Engineering
Instructor: Dr. Charlie Pham

Software Requirements Specification
for
Cryptext
Version 3.0

Prepared By
Team: Chatters, 4

Alejandro Ramos, ramosa1@wit.edu

Ghasif Syed, syedg@wit.edu

Ryan Salazar, salazarr2@wit.edu

02/25/2022

Contents

Revision History	3
Introduction	5
Document purpose	5
Product overview	5
User requirements	5
Definitions	5
Acronyms and abbreviations	5
System requirements	5
Functional requirements	5
Non-functional requirements	7
Other requirements (optional)	8
System architecture	9
Overall architecture	9
Components mapping	9
Functional requirements	9
Non-functional requirements	10
Technology stack selection	10
References	10

Revision History

Date	Version	Description	Author(s)
02/02/2022	1.0	Added the required information on the cover page	Ghasif Syed Alejandro Ramos Ryan Salazar
02/03/2022	1.1	Added introduction content to the document	Alejandro Ramos Ryan Salazar Ghasif Syed
02/03/2022	1.2	- Added account creation and sending messages as functional requirements - Added account customization as non-functional requirement.	Alejandro Ramos
02/03/2022	1.3	- Added user requirement for managing accounts - Added 2 functional REQ: User shall receive email upon account creation, User shall be able to change password - Added 1 non-functional REQ: Account shall be locked upon 10 failed login attempts	Ghasif Syed
02/04/2022	1.4	- Added legislative and ethical requirements	Ghasif Syed
02/04/2022	1.5	- Added user requirement for adding and managing connections - Added 2 functional REQ: User shall be able to add friends, User shall be able to block others - Added 1 non-functional REQ: User shall be to sort connections	Ryan Salazar
02/25/2022	2.0	- Added encryption functional requirement - Merged some functional requirements together - Added components mapping for profile customization and sending encrypted messages	Alejandro Ramos

02/25/2022	2.1	<ul style="list-style-type: none">- Created System Architecture design using Client Server + MVC architecture-Added technology stack-Changed usability non-functional requirement-Added components mapping for the rest of the requirements	Ghasif Syed Alejandro Ramos Ryan Salazar
04/22/2022	3.0	Finalize document with all revisions	Alejandro Ramos Ghasif Syed Ryan Salazar

1. Introduction

1.1. Document purpose

This document provides software requirements specification (SRS) for an online encrypted chat application with the utmost security and safety for clients, along with security for the client we would like to introduce many personalization & QOL features.

1.2. Product overview

1.2.1. Most desktop chat applications are not end-to-end encrypted, meaning that the people who run the chat server can read your messages. This can be a big problem for people because they have to trust that the company or organization isn't reading or saving all of their private messages. Some companies might be using this data in nefarious ways such as advertising products catering to the customer by sifting through personal messages.

1.2.2. Our goal is to develop a chat application that is end-to-end encrypted, ensuring that no one, not even the owners of the chat server, can read the message while it goes from the sender to the recipient. This is allowing for much more privacy than before. The implementation of a complete end-to-end encrypted messaging platform will eliminate the security risk and safety risk that individuals endure on a normal chat client.

1.2.3. Currently most chat clients are not end-to-end encrypted, especially on the computer. Some of the ones that are, do not have it enabled by default or they require additional modules to be installed manually. Our program will have it automatically enabled when the user first installs it. This ensures that the client can be confident in the security of the platform they are using.

1.3. User requirements

The Chat Application shall allow users to:

- Manage account (Ghasif Syed)
- Manage contacts (Ryan Salazar)
- Create accounts and send messages (Alejandro Ramos)

1.4. Definitions

Account: A record of a user's information, such as email address and contact information

Contact: Accounts that a user is friends with, received/sent messages, or has/been blocked from contact

1.5. Acronyms and abbreviations

SRS: Software requirements specification

QOL: Quality of life

REQ: Requirement

2. System requirements

2.1. Functional requirements

2.1.1. Manage account

- [REQ-01] The system shall send a welcome email to user upon account creation

Test description: Send an email when a user creates an account

Precondition: A new user is creating an account

Test steps:

1. User creates an account.

Expected result: The user should receive an email that the new account is created.

- [REQ-02] The system shall allow users to change their password.

Test description: Change an account's password

Precondition: User has an account

Test steps:

1. Go to the change password page.
2. Fill out the required information (current password, new password)
3. Submit

Expected result: Account password has been changed & user can login with new password

2.1.2. Add and manage contacts

- [REQ-01] The system shall allow users to add others as friends.

Test description: Add another account as a friend

Precondition: Two users have separate accounts

Test steps:

1. User clicks on other user's display name.
2. User requests other user to add them as a friend.
3. Other user accepts friend request.

Expected result: Both users are friends with one another and is displayed as such

2.1.3. Create account

- [REQ-01] The system shall allow users to create accounts.

Test description: Create account

Precondition: A user tries to create an account

Test steps:

1. User clicks on "create account" button on login page
2. User enters information such as a username, email address, password, and avatar
3. User clicks "confirm".

Expected result: The user's information is placed in the database with a unique user ID assigned to him.

2.1.4. Send message

- [REQ-01] The system shall allow users to send messages to other users

Test description: Send message

Precondition: There are at least two users with accounts

Test steps:

1. User enters another user's display name or clicks on their name in a listing
2. User is shown a text entry field with their chat history with the other user.
3. User enters a message.
4. User clicks "send" or presses enter.

Expected result: The message is encrypted then sent to the other user.

2.1.5. Encrypt messages

- o [REQ-01] All messages shall be end-to-end encrypted using asymmetric encryption

Test description: Send encrypted message

Precondition: There are at least two users with public/private key pairs

Test steps:

1. User is in the chat window for his conversation with the recipient.
2. User types a message to the recipient and then clicks the send button.

Expected result: The message is symmetrically encrypted using something like AES-256.

The encryption key is asymmetrically encrypted using something like RSA and the recipient's public key. The encrypted message and the encrypted key are then sent through the server to the recipient.

2.2. Non-functional requirements

2.2.1. Security

- o [REQ-01] After 10 login attempts, the system shall lock an account to protect a user's information from potential hackers (safety)

Test description: Security account lockout after 10 failed login attempts

Precondition: User has an account

Test steps:

1. Attempt login with the incorrect password x10
2. Attempt to login again
3. Submit

Expected result: User should be prompted with an account lockout screen along with steps on account recovery

2.2.2. Usability

- o [REQ-01] The system shall allow the user to find friends in two clicks (usability)

Test description: Find friends within two clicks

Precondition: User has an account and has logged in

Test steps:

1. User clicks on their friend list
2. User types in their friends username
3. User clicks on the search button

Expected result: User should see their friends username

2.2.3. Customize profile

- o [REQ-01] The system shall allow users to change their username and password

Test description: Change avatar/username

Precondition: User has an account

Test steps:

1. Go to the settings page
2. Enter a new username or password
3. Click confirm

Expected result: The user's information is updated in the database and sent to the other users if they are friends with them or in a conversation with them.

2.3. Other requirements (optional)

2.3.1. Legislative

- [REQ-1] The system shall have terms and conditions on privacy of user data.

3. System architecture

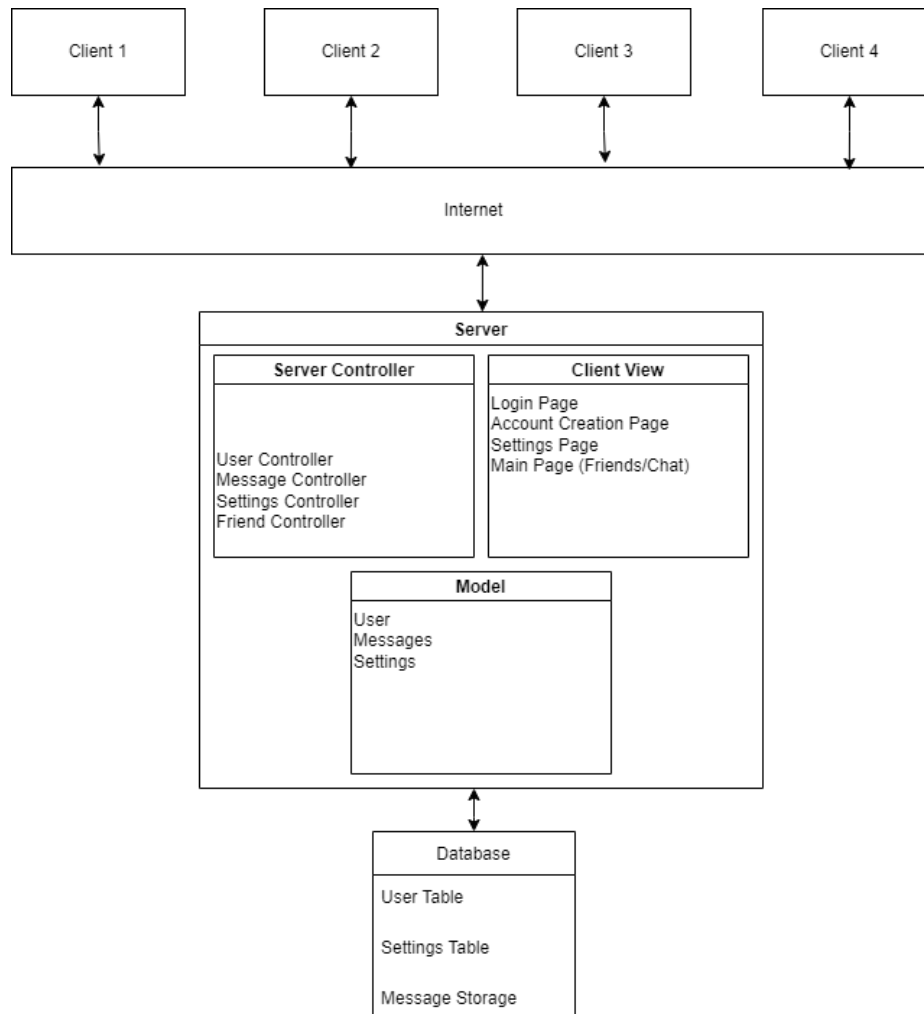


Figure 1. System architecture of Cryptext. Uses client-server and MVC

3.1. Overall architecture

Our system will be using a mix of client-server architecture and MVC architecture. Client-server will allow multiple users to access our server(s) to message one another. The server itself will be based on the MVC architecture. Users will have pages that they can view and interact with which alert the necessary controllers to update the model. The model will grab whatever information it needs from the database so that the user's view can update accordingly.

3.2. Components mapping

3.2.1. Functional requirements

Manage Account

- Client interacts with Settings Page
- Settings Page interacts with Settings Controller
- Settings Controller interacts with Settings model

Add and Manage Contacts

- Client interacts with Main Page
- Main Page interacts with Friend Controller
- Friend Controller interacts with Friend model, which handles all contacts

Create account

- Client interacts with Account Creation Page
- Account Creation Page interacts with User Controller
- User controller interacts with User Model

Send encrypted messages

- Client Interacts with Main Page
- Main Page interacts with Message Controller
- Message Controller interacts with Messages Model

3.2.2. Non-functional requirements

Security

- Client interacts with Login Page
- Login Page interacts with User controller
- User Controller interacts with User model

Usability

- Client interacts with Main Page
- Main Page interacts with User and Friend Controller
- Controller interacts with User and Friend Models

Customize Profile

- Client interacts with Settings Page
- Settings Page interacts with Settings Controller
- Settings Controller interacts with Settings Model

3.3. Technology stack selection

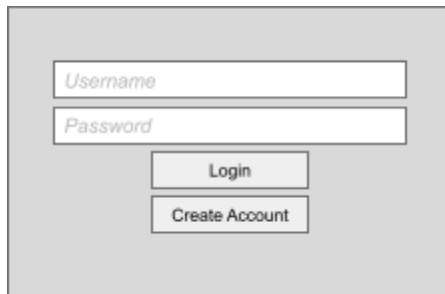
- Front end - Java + JavaFX
 - Chosen because it is built into Java and easy to work with when using the Gluon Scene Builder to generate the UI.
 - Client UI
- Back end - Java
 - Chosen because the team has lots of experience with it, especially the network sockets. It also has built in libraries for the cryptographic functions our project requires.
 - Built in RSA library
 - Built in AES
 - Network sockets
- Database - Postgres

- Chosen because Postgres is a very popular database tool and because our team has experience using it in both web and standalone applications.
- Relational Database

4. System Design

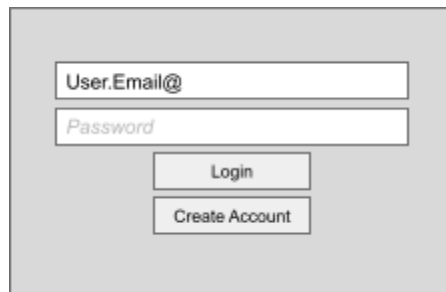
4.1. UI

4.1.1. Create Account



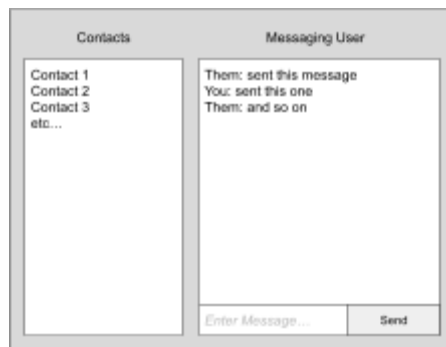
A UI form for creating an account. It features two input fields: the top one is labeled 'Username' and the bottom one is labeled 'Password'. Below these fields are two buttons: 'Login' and 'Create Account'.

4.1.2. Sending Welcome Email



A UI form for sending a welcome email. It features two input fields: the top one is labeled 'User.Email@' and the bottom one is labeled 'Password'. Below these fields are two buttons: 'Login' and 'Create Account'.

4.1.3. Send Messages

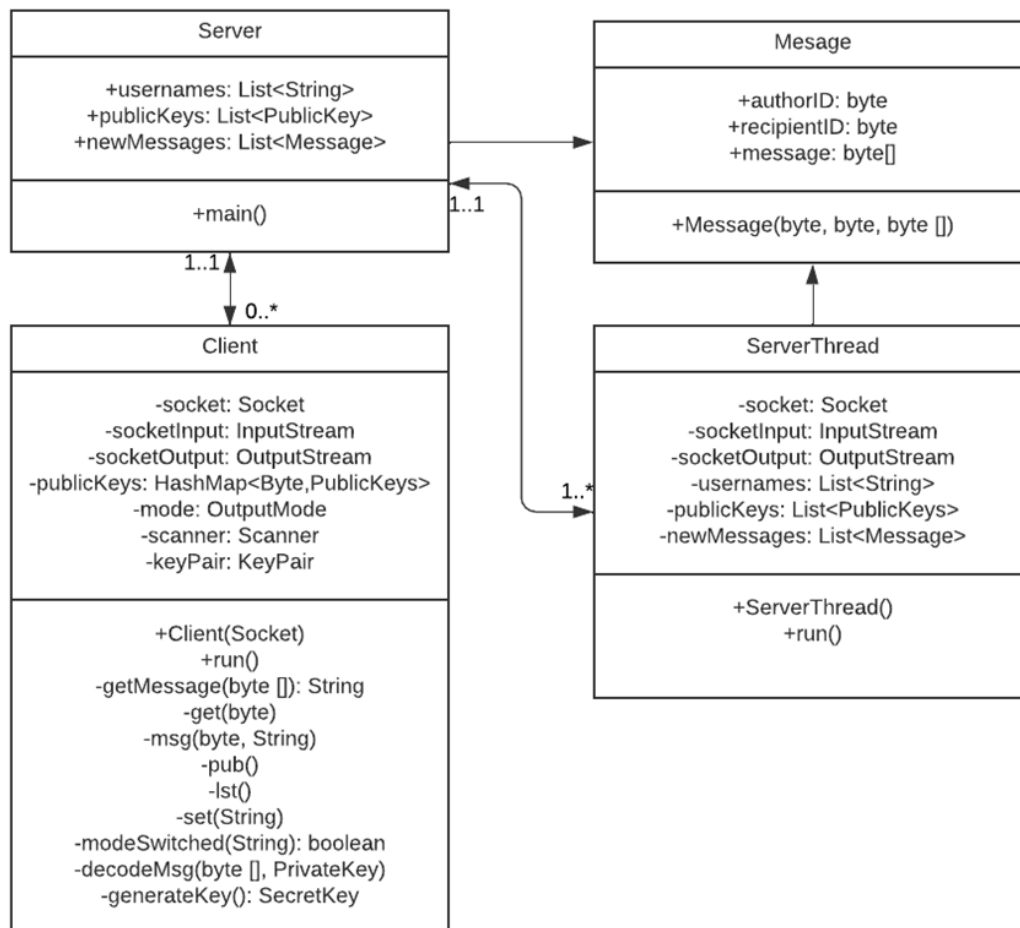


A UI form for sending messages. It is divided into two main sections: 'Contacts' on the left and 'Messaging User' on the right. The 'Contacts' section contains a list of contacts: 'Contact 1', 'Contact 2', 'Contact 3', and 'etc...'. The 'Messaging User' section contains a text area with the following text: 'Them: sent this message', 'You: sent this one', and 'Them: and so on'. Below the text area is a text input field labeled 'Enter Message...' and a 'Send' button.

4.2. Class Diagram

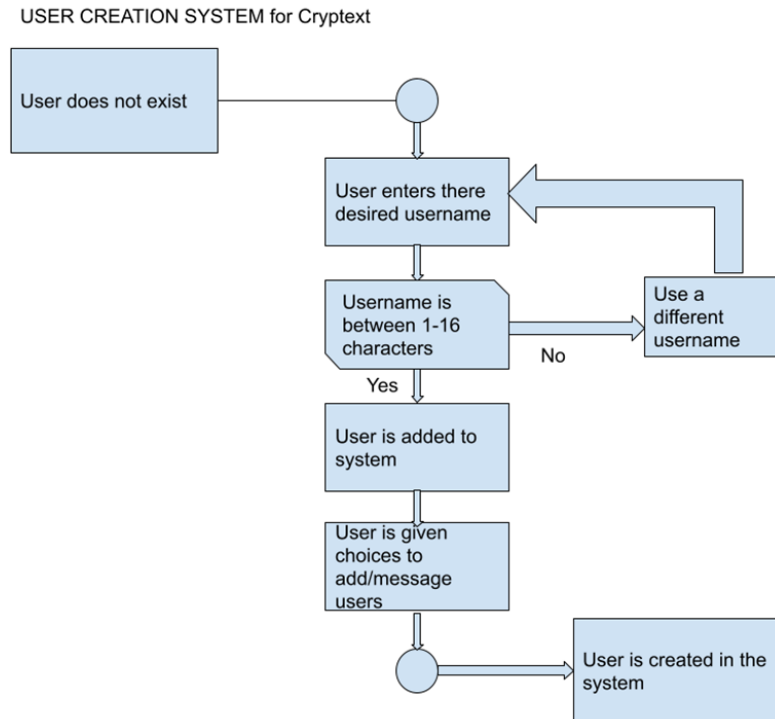
UML class

Ghasif Syed | March 27, 2022

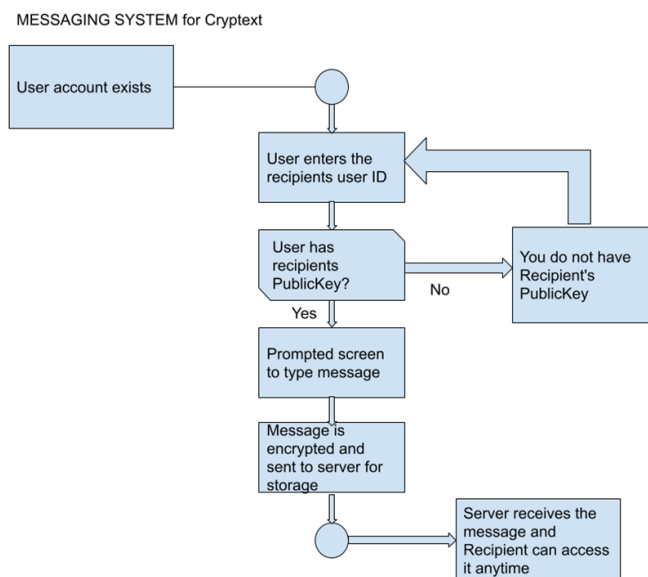


4.3. Sequence/Activity Diagram

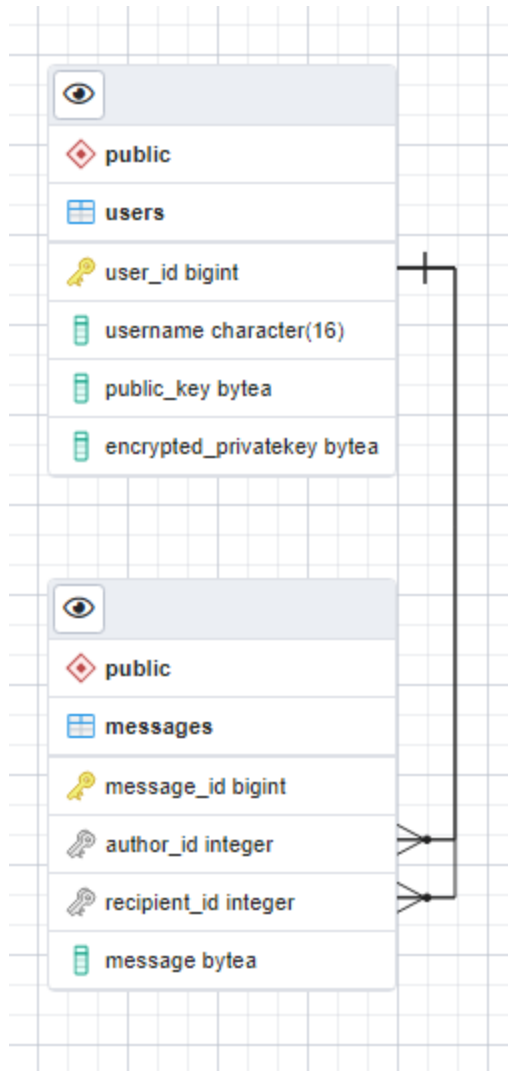
4.3.1. Create User



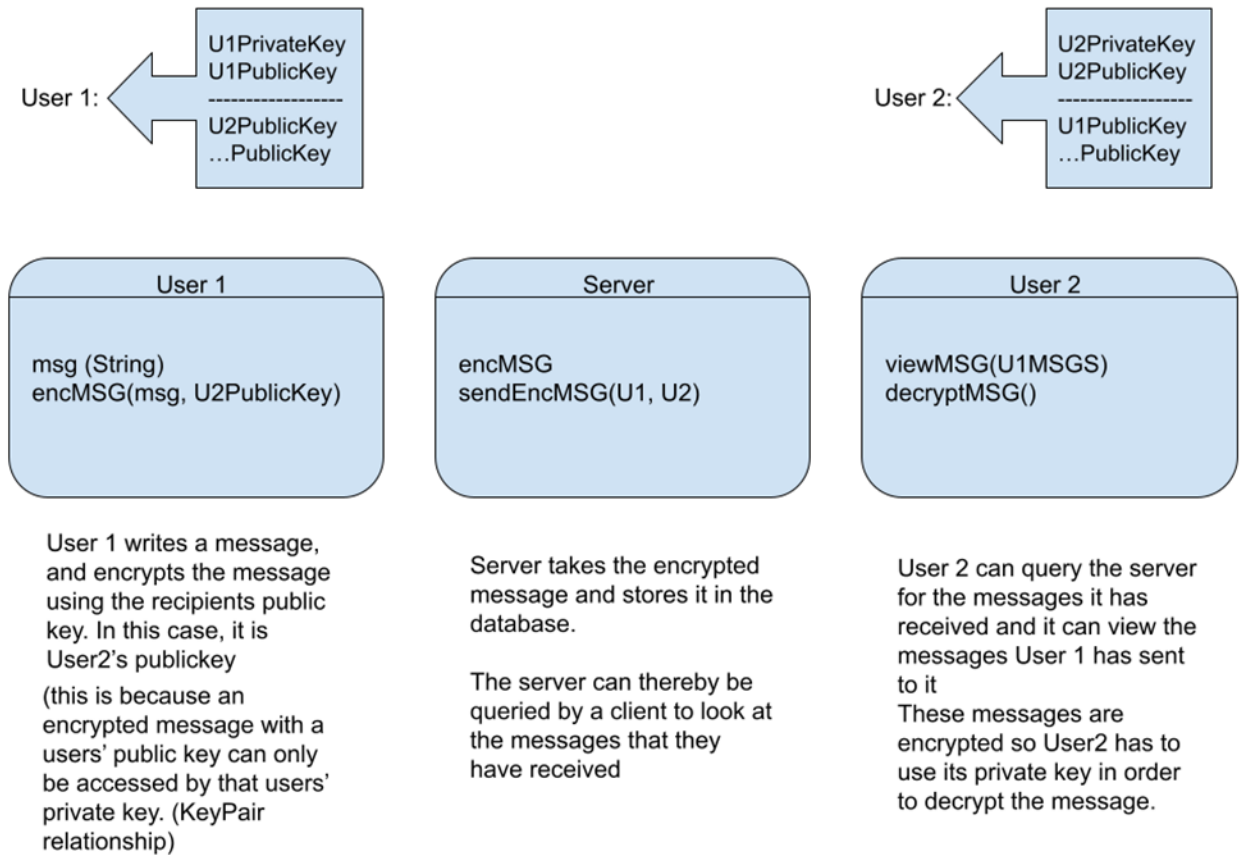
4.3.2. Sending a Message



4.4. Database



4.5. Encryption Diagram



4.6. Protocol

The client and server communicate using a custom protocol that we designed.

LST	returns list of users and their IDs
PUB	returns the public key of the given user
MSG	sends a message to the given user
GET	returns the conversation between the sender and the given user
LOG	returns Y if the login was successful, N if the given username does not have an account
REG	registers a new account with the given username, public key, and encrypted private key. returns their new ID

NME	renames a user. returns S if the rename was successful, or F if unsuccessful. two users can not have the same username
PWD	changes a user's password. returns S if the signature on the request matches the public key in the user's file. returns F otherwise.

5. Others

None

6. Test Plan

6.1. Create Account

No.	Test Case	User Input	Pass Criteria
1	Create account	username: test password: test1234	New user is created
2	Create account with empty fields	No user input	Display message: "Enter desired Username and Password, then click this button."
3	Create account with username longer than 16 characters	Any username longer than 16 characters, such as: longerthansixteen Any password	Display message: "Invalid length. Please try again."
4	Create account with username starting with "\"	Any username starting with "\" Any password	Display message: "Username can not start with '\'. Please try again."

6.2. Send Message

No.	Test Case	User Input	Pass Criteria
1	Send a non-empty message	test message	Message is sent and displayed
2	Send an empty message	No user input	Nothing is sent

7. References

<https://www.baeldung.com/java-rsa>

<https://www.baeldung.com/java-aes-encryption-decryption>