*Chat system with secure communication*

*September 13, 2023*

[Introduction 2](#_Toc379904795)

[Team Resume Filler: 3](#_Toc764261430)

[Milestone 1: 4](#_Toc1163401112)

[System Requirements 4](#_Toc1774559650)

[Requirements: 4](#_Toc819015458)

[User Stories: 4](#_Toc535767870)

[Project Management 4](#_Toc1419710463)

[Continuity of Operations Plan (COOP) 5](#_Toc1865408852)

[Project Plan: 5](#_Toc2105031720)

[System Architecture Design and Development 6](#_Toc1864353803)

[Milestone 2 (Architecture): The WBS activity chart for the milestone should be updated to include actual level of effort and start and completion dates.] 6](#_Toc1821441212)

[System Implementation <Milestone 2: Architecture & Milestone 3: System Implementation> 6](#_Toc1001699862)

[Project Postmortem <Postmortem> 6](#_Toc920124337)

[Project Wins 7](#_Toc289981942)

[Root Cause Analysis 7](#_Toc233040043)

[Lessons Learned 7](#_Toc43930894)

[System Design <Milestone 2: System Architecture> 7](#_Toc1143018302)

[System Architecture <Milestone 2: System Architecture> 8](#_Toc501350651)

[Component Design 8](#_Toc104202792)

[Data Flow 8](#_Toc1101335783)

[System Components <Milestone 3: System Implementation> 8](#_Toc1545052108)

[Component [Component Name 1] 8](#_Toc850514367)

[Component [Component Name 2] 8](#_Toc1241188815)

[Component [Component Name n] 8](#_Toc566185563)

[Design Pattern <Milestone 3: System Implementation> 8](#_Toc2099398861)

[Design Pattern <Milestone 3: System Implementation> 8](#_Toc1645943909)

[System Implementation <Milestone 3: System Implementation> 9](#_Toc1379425202)

[Project portfolio template directives and placeholders (delineated by “[ ]” or “< >” and/or highlighted or optional sections not included) should be removed from the document prior to submission. Empty sections for inclusion in later submissions may remain in the document for early submissions.]

[IMPORTANT: All diagrams developed using Enterprise Architectures must include the following acknowledgement: “Thanks to SPARX Systems for LSU student and faculty use of Enterprise Architect for academic purposes”.]

# Introduction

When connecting over the internet and sending any form of data, it is important to be able to trust that the information being sent is safe from any outside influence. Privacy is important to all users, and critical to some. A solution is to have a trusted tool to connect to a remote machine that allows users to send and receive data that is automatically encrypted before sending and decrypting after receiving.



Core Features:

* Sending encrypted messages
* Decrypting received messages
* Direct secure communication between machines

Viable Features:

* GUI (through C++, subject to change)
* File encryption, decryption, and transfer

Stretch Features:

* Adding friends
* Group chat
* Profile System
* Message reactions

# Team Resume Filler:

## Milestone 1:

Leader: Abby Debenport

Members: Arabelle Betzwieser, Kenyon Tiner, Tyler Saizan, Arushi Ghildiyal, Connor French

## Milestone 2:

Leader: Arushi Ghildiyal

## Milestone 3:

Leader: Arushi Ghildiyal

[GitHub Link](https://nam04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fclassroom.github.com%2Fa%2FUCqQgtmZ&data=05%7C01%7Cktiner1%40lsu.edu%7C844c2d154411489edd6808dbabaf3521%7C2d4dad3f50ae47d983a09ae2b1f466f8%7C0%7C0%7C638292541286440561%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=5GDmBWTIHr7WyDUmVVPbO4bmESBiqjZN5Ll8B2YcI08%3D&reserved=0)

# System Requirements

## Requirements:

-Socket Programming

-Encryption/Decryption

-Communication between machines

-GUI implementation

## User Stories:

As a student, I want to form a group chat with my group mates, so I can communicate privately with them about our project.

As a programmer, I want to directly and securely send files to my team, so they can assist with my code.

As a team leader, I would like to see reactions to my messages, so that I have confirmation my message has been received.

As an investigator, I want to send encrypted investigation reports to my supervisors, so that the information is not leaked to the public.

# Project Management

## Continuity of Operations Plan (COOP)

* The team will primarily use a discord private server to communicate with each other and coordinate. Email will be a secondary means of communication.
* The team will meet for 30 minutes each Tuesday after class and on Sundays at 2pm over discord.
* If team members are unable to meet in-person or become unavailable, the team will hold virtual calls and update missing members through discord and/or email.

## Project Plan:

### System Architecture Design and Development

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Activity** | **Pre #** | **Estimated**  **Effort** | **Actual**  **Effort** | **Estimated**  **Start Date** | **Estimated**  **Finish Date** | **Actual**  **Start Date** | **Actual**  **Finish Date** |
| **1** | Create Discord server for main method of comm. | 0 | low | low | 9/12/23 | 9/12/23 | 9/12/23 | **9/12/23** |
| 2 | Initial Individual research on the topic | 0 | low | low | 9/13/23 | 10/03/23 | 9/13/23 | 10/01/23 |
| 3 | Identify individual tasks and delegate | 1 | medium | high | 9/16/23 | 9/20/23 | 9/14/23 | Continued |
| 4 | Initial consultation with TA | 0 | low | low | 9/12/23 | 9/13/23 | 9/12/23 | 9/13/23 |
| 5 | Postulate application design |  | high | high | 9/17/23 | 9/22/23 | 9/17/23 | 9/23/23 |
| 6 | Begin Iterative development process | 5 | high | high | 9/19/23 | continuing | 9/17/23 | Continued |

### System Implementation

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Activity** | **Pre #** | **Estimated**  **Effort** | **Actual**  **Effort** | **Estimated**  **Start Date** | **Estimated**  **Finish Date** | **Actual**  **Start Date** | **Actual**  **Finish Date** |
| 1 | Networking: Build server skeleton | 0 | High | low | 10/10/23 | 10/17/23 | 10/10/23 | 10/14/23 |
| 2 | Networking: Build client skeleton | 0 | medium | low | 10/10/23 | 10/17/23 | 10/10/23 | 10/14/23 |
| 3 | Encryption: Encrypt messages | 0 | medium | medium | 10/10/23 | 10/17/23 | 10/12/23 | 10/24/23 |
| 4 | Encryption: Decrypt messages | 3 | medium | medium | 10/10/23 | 10/17/23 | 10/12/23 | 10/24/23 |
| 5 | Networking: Authentication | 1, 2 | medium | low | 10/13/23 | 10/19/23 | 10/14/23 | 10/16/23 |
| 6 | Networking: Setting up client key-grabbing interface | 2 | medium | medium | 10/13/23 | 10/19/23 | 10/20/23 | 10/25/23 |
| 7 | Networking: Message handling | 1, 2 | medium | medium | 10/10/23 | 10/17/23 | 10/14/23 | 10/26/23 |
| 8 | Basic functional demo | 1-7 | High | high | 10/10/23 | 10/22/23 | 10/10/23 | 11/10/23 |

## Project Postmortem <Postmortem>

### Project Wins

[Provide a bulleted list of at least 3 positive aspects of the project.]

### Root Cause Analysis

[Provide a bulleted list of at least 3 negative aspects of the project. For each negative, provide the answer to the three successive “Why” questions. ]

### Lessons Learned

[For each negative aspect identified in the Root Cause Analysis, provide a mitigation strategy (i.e., what process should be introduced) to ensure that the problem is not repeated in subsequent projects.]

# System Design

A client-server architecture with a cryptology interface on the client-side, enabling safe and secure communication between clients.

## System Architecture

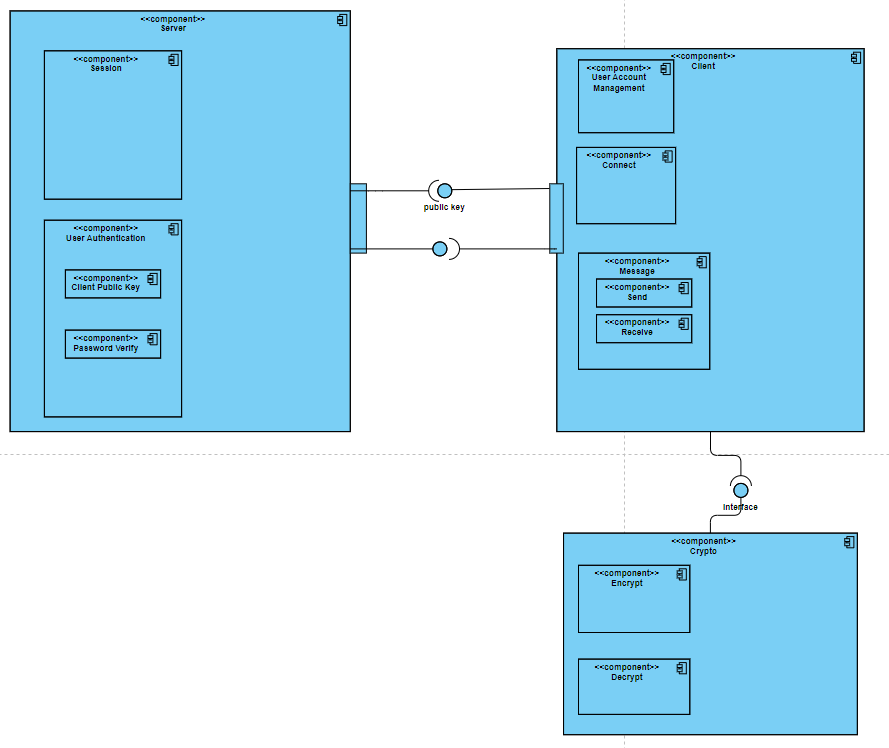
Mailman server directs messages to and from clients.

Clients encrypt outgoing messages and decrypt incoming messages through communication with server.

Cryptography module handles decryption/encryption and key generation.

### Component Design

Abby Debenport:

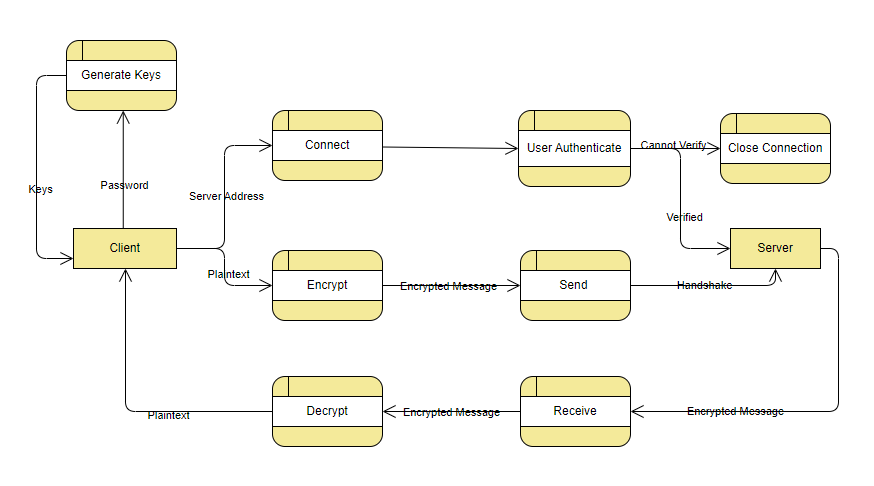
User’s cryptography tasks are handed off to cryptography module. User inputs messages, receives messages, not having to directly handle encryption process. Server only passes already encrypted messages between users.

External libraries include: libsodium, Asio

Major components: Server, Client, Cryptography module

### Data Flow

Abby Debenport:

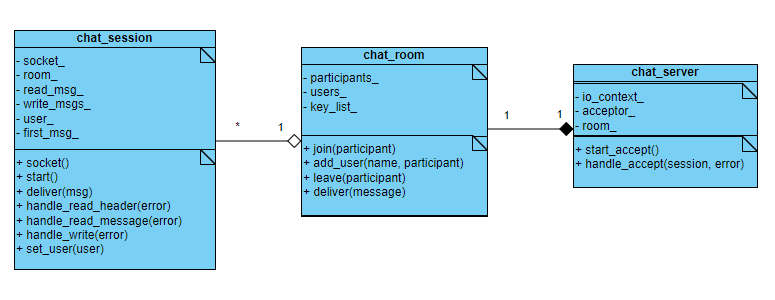
User Inputs message into client-side application where message is encrypted before being sent to the server. The server passes the encrypted message to another client using a public key; once the message is received by the client, the message is decrypted showing the original message.

## System Components:

## Component Server

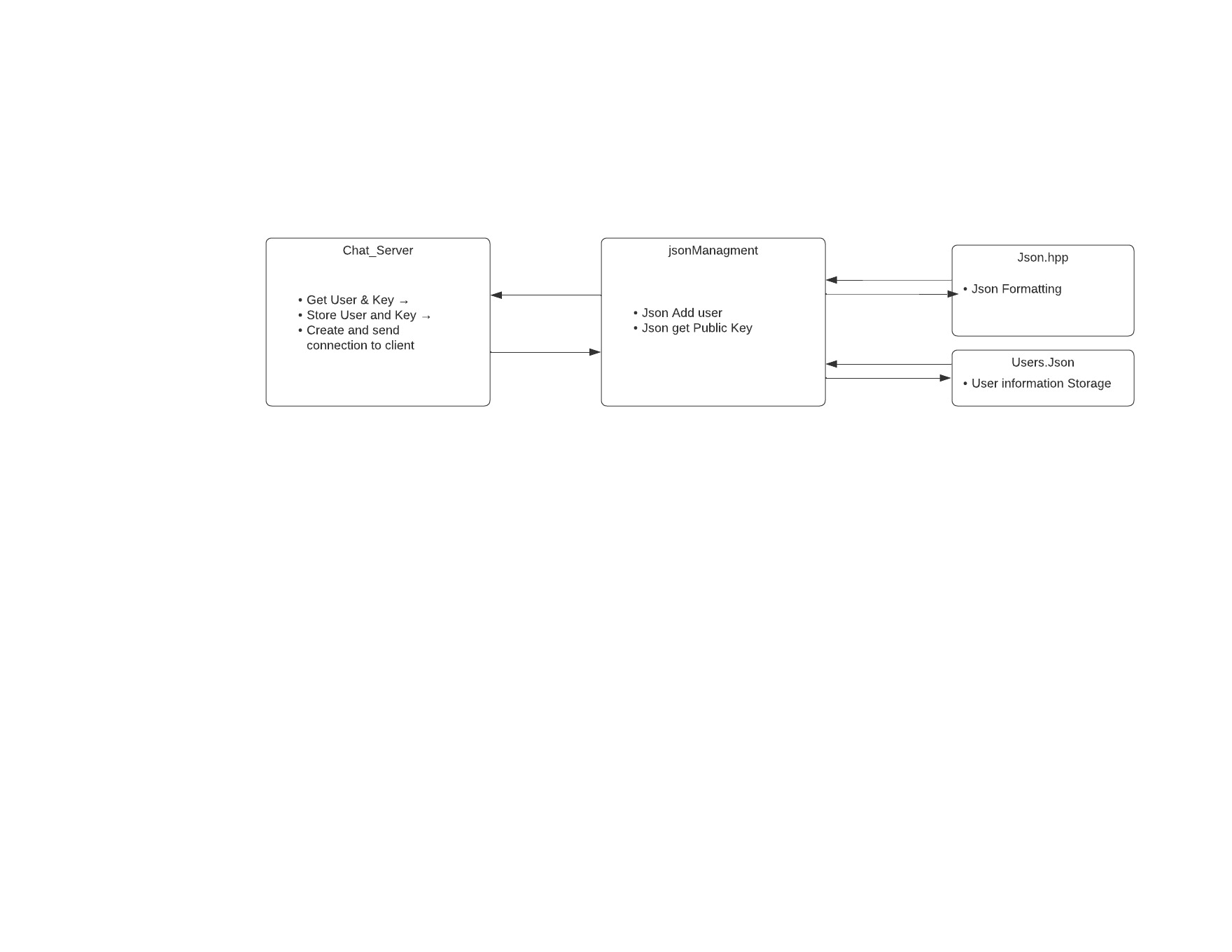
The server handles the routing of messages to the appropriate clients, as well as storing & sending out key pairs to newly connected clients.

Arabelle Betzwieser:



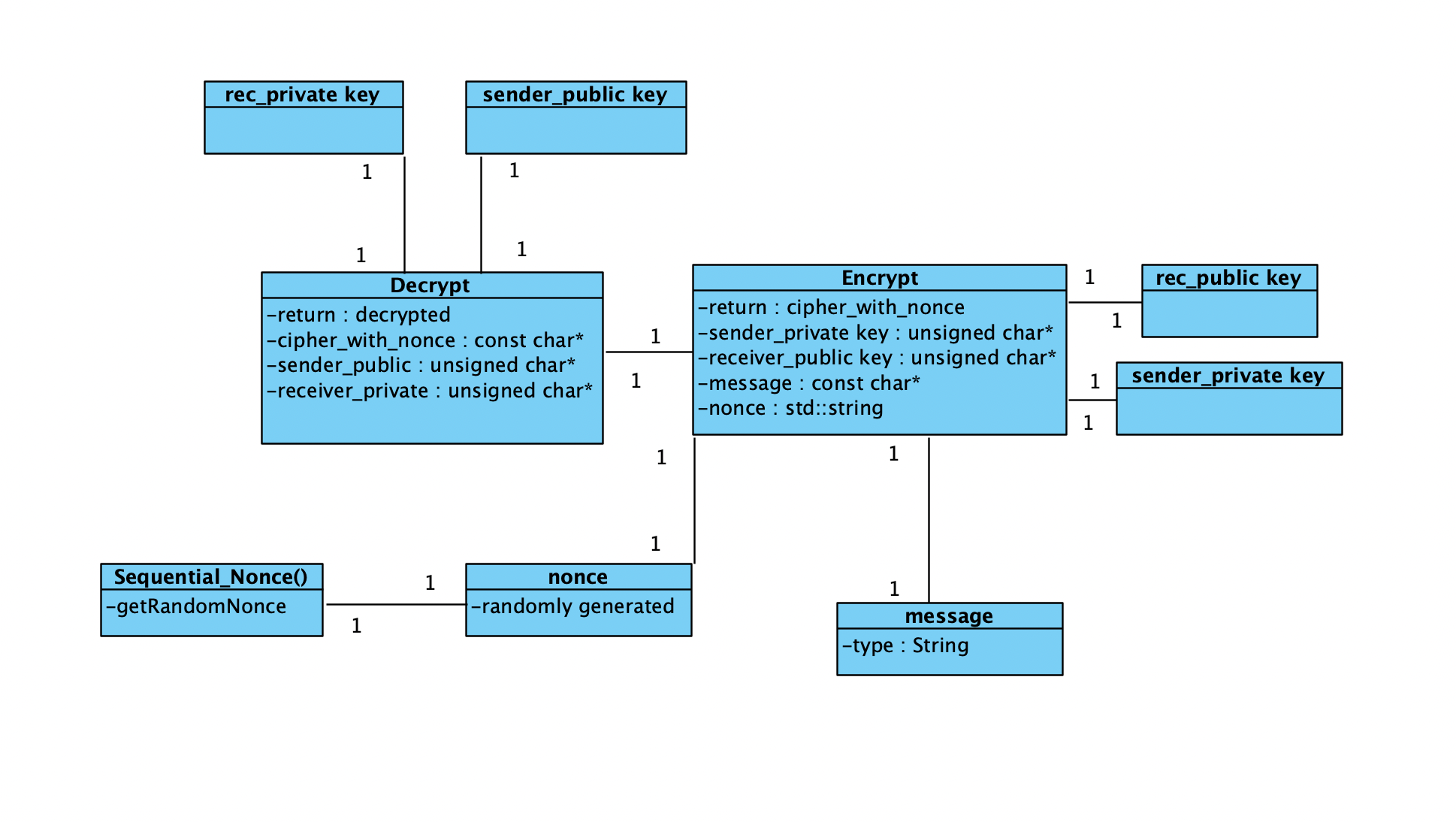
### Component Databasing - Arushi Ghildiyal

To hold the non-sensitive information of a user we used a Json file to hold the public keys and usernames associated with each user.

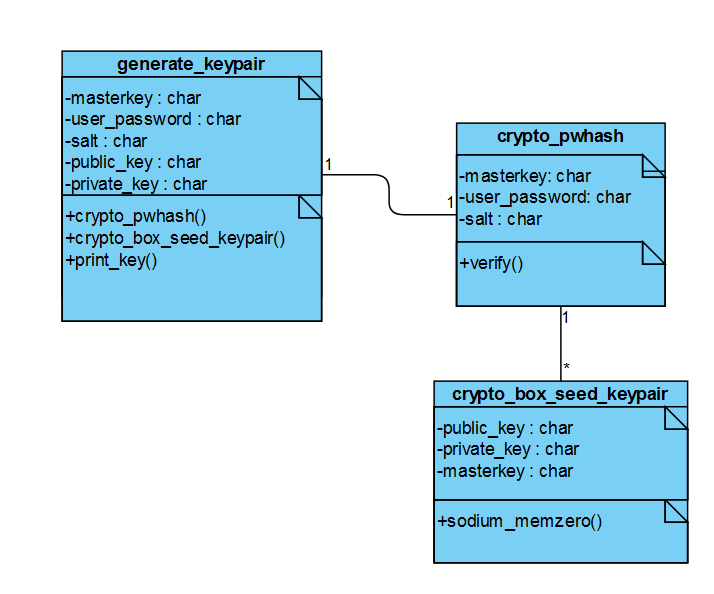


### Component Encryption/Decryption

The encryption/decryption component handles encryption and decryption of user messages and sends decrypted messages to authorized recipients that hold the correct key pair.

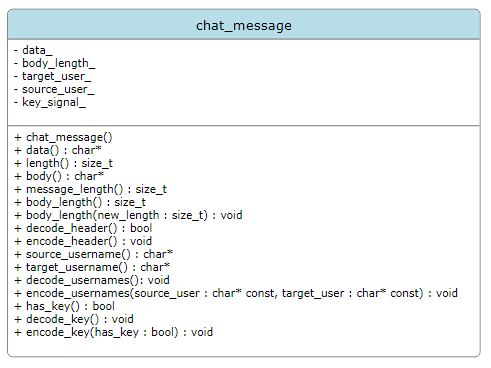
Kenyon Tiner:  


Connor French:



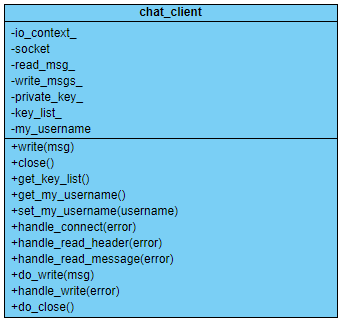
### Component Message – Tyler Saizan

The message class provides data members and functions for a message object, which is handed between the client and server in each interaction.



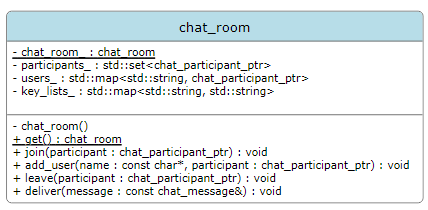
### Component Client – Abby Debenport

The client connects to the server and logs in with username and password. Then the client can read and write messages to and from the server.



## Design Pattern Singleton – Tyler Saizan

Only one chat room is run on a chat server, so it was a candidate for the Singleton design pattern. This ensures that no more than one instance of the chat\_room class is created in a single server and provide easy access to that instance.



# System Implementation

|  |  |  |
| --- | --- | --- |
| **Architectural Component** | **Programming Language(s) %** | **Team Member(s) %** |
| *Database* | *[C++ (100%)]* | *Arushi Ghildiyal (100%)* |
| *Encryption/Decryption* | *[C++ (100%)]* | *Kenyon Tiner (80%)*  *Connor French (20%)* |
| *Key Gen* | *[C++ (100%)]* | *Connor French (100%)* |
| *Server* | *[C++ (100%)]* | *Arabelle Betzwieser (60%)*  *Abby Debenport (40%)* |
| *Client* | *[C++ (100%)]* | *Arabelle Betzwieser (30%)*  *Abby Debenport (40%)*  *Tyler Saizan (30%)* |
| *Message* | *[C++ (100%)]* | *Tyler Saizan (80%)*  *Arabelle Betzwieser (20%)* |