Logbook

SDD HSC Major Work - Messenger
Max Hamilton

Table 1: Log Book

Date	Summary
22/10/2022	Came up with idea for the project. A messenger app. I put a bit of thought into how it would work, and the client. The client would be a company that needs a secure messaging service similar to email. I would provide a client software, and a server software. The client software would be a GUI, and the server may be a GUI or a CLI. The server software will not be able to read messages, and will have the minimum possible information about the messages. This allows for secure outsourcing of the server hosting if the company cannot run their own servers. To add a new user, the server would generate a code which the system administrator could communicate to the new user, which the client uses to send a username and password to the server. On initialisation, the client would generate a public and private key. The client keeps the private key, and sends the public key to the server. To send a message, the client requests the recipient's public key, and encrypts the message with it. The encrypted message is then sent to the server, where it is stored indefinitely. I think I can make it anonymous who exactly is sending and receiving messages. The message on the server is then available for download by all clients, but only the recipient can use their private key to decrypt it. I can have anonymous uploads by encrypting the name of the sender with the rest of the message, and anonymous downloads by having a constant magic number at the start of each message. Every client, on seeing a new available message, would try to decrypt it with their private key, and if the constant magic number didn't match, they could discard it, meaning only the recipient would be able to access it. Additionally, messages would be signed with the sender's public key, then the signature encrypted with the rest. That way, once the message was decrypted, the identity of the sender could be verified. A bad actor could download all encrypted messages, but the would not know the intended recipient, sender, or content. A lot of aspects could be controlled by the server software, such as file size caps, message size limits, and length of saved history.
22/10/2022	Formulated my thoughts by writing the description of the solution, in a lot of detail. Might have to cut it down later.

Table 1: Log Book (Continued)

22/10/2022	I have finished the first draft of the design brief, and have a complete plan for the high-level design of my program.
23/10/2022	I have done some research and begun to implement the core functionality of the client. Experiencing some difficulties deciding to encrypt files on disk, or to pull them into memory.
29/10/2022	I have located the cryptography libraries to use, as well as encrypting files bit by bit so they do not take up huge amounts of memory.
16/11/2022	I have written the code for clients sending messages, and the server receiving and storing them.
23/11/2022	I have written the code for clients asking about the recipiency of messages, and the server answering.
30/11/2022	I have written the code for clients downloading message contents.
05/2/2022	I have started writing test code for the general sending and receiving of messages.
10/2/2022	I have finished writing the test code, and am debugging numerous issues relating to encryption, error handling, and bugs in test code.
16/2/2022	I have gotten single-threaded tests working, and am now writing multi-threaded tests.
20/1/2022	I have added mutex-style guards on the files storing messages, which should eliminate most multi-threaded bugs.
25/11/2022	I have finished debugging the multi-threaded code, and the full test suite is passing without bugs.