Flaws in previous version

My previous program contained no form of encryption for the data being sent between the client and server. This left it vulnerable to attacks from unauthorized people.
The client and server programs also did not contain any form of authentication to stop unauthorized users from accessing the programs.

Solutions

To solve the first problem of the encryption, I implemented RSA 3072 bit encryption for both the Client and Server programs to encrypt any data being sent between them. This is covered in more detail in the ‘protocol.pdf’ file. In short both a private key and public key are generated for both the client and server. They both exchange their public keys upon connecting to each other so they can encrypt and decrypt data that is exchanged

For the problem of ensuring that the client is authorized to use the server, I thought that the simplest fix for this would be to add a password to the client program that only authorised users would have access to. This password is stored using SHA256 hashing and is also salted to provide further security against unauthorized users. As the server will no longer work with the previous client program from the last assignment, it means that only users that have access to the updated client program with the password will be able to access the program. The client also implements a timeout function - if 5 incorrect passwords are entered the client program will close to help protect against brute force attacks against the password.

Potential flaws

If an attacker is able to gain access to the password stored in the ‘pass.txt’ file through some form of attack it could give them access to the whole program as there are no further authentication checks once the client has entered the correct password.

The program could also potentially be brute forced if the client chooses an insecure password, as there is only a basic method of closing the program if 5 incorrect passwords are entered to stop this. The best way to stop this is ensuring that the user chooses a secure password that contains a mix of letters in lower and uppercase, numbers, special characters and is more than 10 characters long

The client could be reverse engineered and used to gain access to the server without having access to the password required to use it.

These problems could be mitigated by adding another password check once the client connects to the server, but for the scope of this project I did not think that it would be necessary as these methods would still be quite time consuming.