SOFE 4790U: Distributed Systems (Fall 2024)

Assignment #1

Honour code: By submitting this assignment, I (name and banner id# below) affirm this is my own work, and I have not asked any of my fellow students or others for their source code or solutions to complete this assignment, and I have not offered my source code or solutions for this assignment to any of my fellow students.

Name: Kramptj KC

Banner ID#: 100787909

1. Application idea

The program for the assignment was to create a file transfer program using the server client model. The application has a directory (resources directory from the project) acting as a pool from where it lists down files within the directory. This directory can be altered to have any files within it. From the client terminal all the files within the directory are listed for the user to upload. Then the file gets encrypted using a password plus a randomly generated 16 byte salt (random/filler) values and gets sent to the server. The server then uses the file name, password and salt values sent by the client to decrypt the file. The encryption algorithm used for this program is AES (Advanced Encryption Standard). The server has its own directory where it stores all the received files it gets from the client and then the encrypted and decrypted files are created and stored in their own respective directories.

2. Describe the two core functionalities.

1. **Encrypted File Transfer**
   The main function of this program is to transfer files securely from the client to the server. Files in this program get encrypted in the client side before getting transferred and then decrypted in the server side. Having encryption means that the data is safe during transmission.

2. **Asynchronous Multithreaded Server**
   The server is multithreaded which means that it can handle multiple clients simultaneously. This allows for file transfers without causing any disruptions/blocking which then means the server is scalable and can be used in a multi-user environment.

3. Describe the two novel features.

1. **Real-time notifications for transfer completion**
   Clients get real-time updates about their file status being transferred. They get to know where it got transferred to and the name of the file after it has been encrypted/decrypted

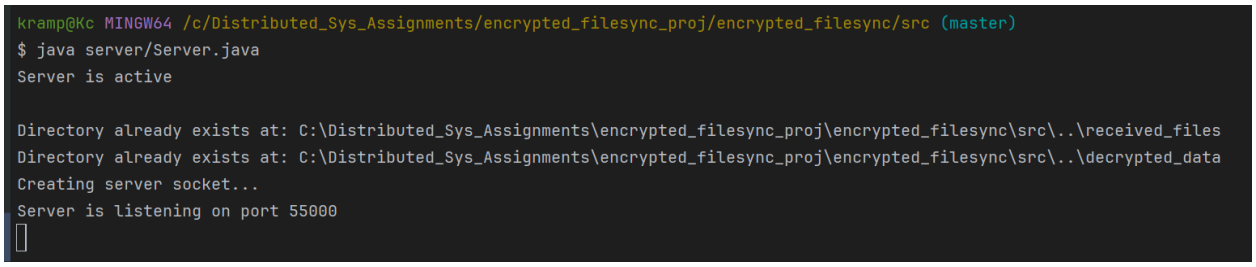
2. **Uses PBKDF2(Password Based Key Derivation Function) algorithm with HMAC-SHA256 (Hash-based Message Authentication Code) hashing algorithm for key generation**
   The program accepts a user-inputted password as well as using the generateSalt() function within the code to generate a 16 byte value. The password and the 16byte value are used to create the secret key which gets sent to the server to match so it can start decrypting the data.

4. Challenges and solutions

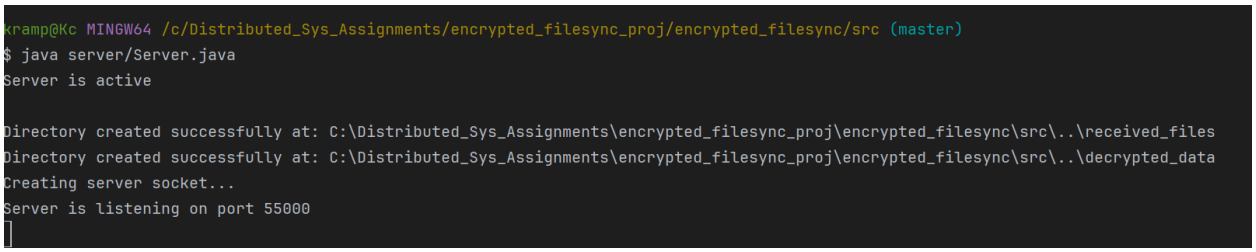
1. **Managing Socket Lifecycle and Connection State**
   Solution: The client and server sockets need proper handling to make sure that they close properly after file transfers. This is challenging to accomplish in a multithreaded environment where connections must remain open for notifications without prematurely closing. So using **try-with-resources** statements carefully, allows each socket to close only after all necessary communication has completed

2. **Ensuring Secure Key Management**
   Solution: The program uses password-based encryption, however managing the security of passwords keys is crucial. Exposing passwords in plain text or reusing the same salt for multiple transfers could lead to the system being vulnerable. So the program **generates a unique salt** for each file transfer to ensure that even with the same password, each encryption key is different.

5. Testing



Server running, Directory already exists



Server running, directory gets created if it doesn't exist

```
kramp@Kc MINGW64 /c/Distributed_Sys_Assignments/encrypted_filesync_proj/encrypted_filesync/src (master)
$ java client/Client.java
Available files to send to server:

- binaryfile.txt
- emptyfile.txt
- largefile.txt
- testfile.txt
Enter the name of the file you want to send:
```

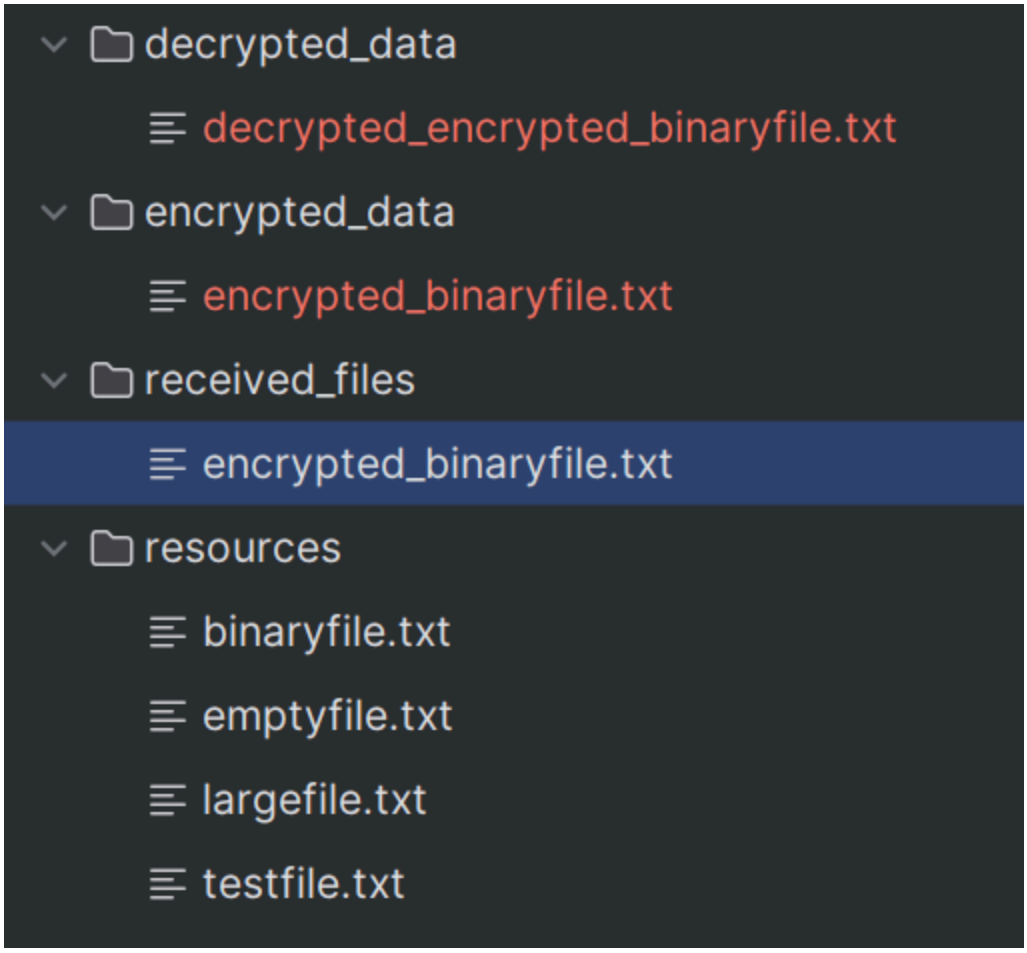
Client needs to input the file of their choice

```
Enter the name of the file you want to send:
binaryfile.txt
Chosen file to encrypt: binaryfile.txt


Enter a password for encryption:
teststsets
Directory 'encrypted_data' created successfully at: C:\Distributed_Sys_Assignments\encrypted_filesync_proj\encrypted_filesync\src\..\encrypted_data
File encrypted successfully as: encrypted_binaryfile.txt
Encrypted file sent: encrypted_binaryfile.txt
Connection to server disconnected.

kramp@Kc MINGW64 /c/Distributed_Sys_Assignments/encrypted_filesync_proj/encrypted_filesync/src (master)
$
```

Enter the name and the program outputs the file path of where it gets encrypted

```
Server is listening on port 55000
Client connected.
Received password from client: teststsets
Received encrypted file name: encrypted_binaryfile.txt
Received salt value: f034e4dbfcb4d4757b099b8f07ed72ea
Encrypted file received and saved at: C:\Distributed_Sys_Assignments\encrypted_filesync_proj\encrypted_filesync\src\..\received_files\encrypted_bina
ryfile.txt
File decrypted successfully at: C:\Distributed_Sys_Assignments\encrypted_filesync_proj\encrypted_filesync\src\..\decrypted_data\decrypted_encrypted_
binaryfile.txt
```

Server has the path to the where the file was received and decrypted location and outputs password from client, salt and file name



Once the files have been encrypted/decrypted, they get placed in their respective directory.