1 Introduction

This project is a security application built on top of the DES implementation in project 1. The topic selected is similar to the given topic: Secure communication. The main goal of this application is to encrypt the data transferred between two users who are engaging in an interactive session of data exchange. Through a user interface, users can send and receive message (or files) which are performed independently from each other during which the data (or files) must be encrypted using DES while in transmission.

2 Architecture, Design and Implementation

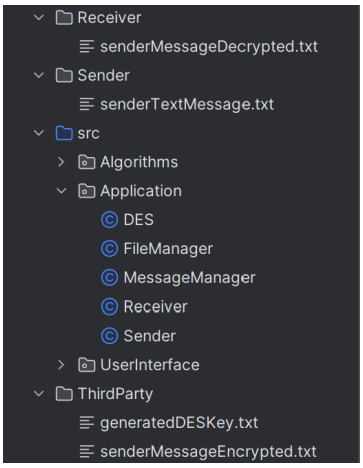

Figure 1: Project code Structure

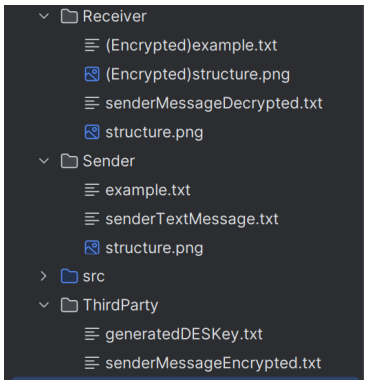

Figure 2: Project code Structure (after file transmissions)

As displayed in figure 1, the DES-based secure communication application is developed based on project 1, i.e pacakges ***Algorithms*** and ***UserInterface***. The application source code consists of classes: ***DES***, ***FileManager***, ***MessageManager***, ***Receiver***, ***Sender***, as well as three information folders ***Sender***, ***Receiver***, ***ThirdParty***.

1. ***MessageManager*** includes:
   - *getEncryptedMessage* that reads and returns the encrypted sender message from *senderMessageEncrypted.txt* in the *ThirdParty* folder.
   - *storeMessage* that stores the message string user sends to *senderTextMessage.txt* in the *Sender* folder.
   - *getRandomString* that generates a random string of a given length with letters "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789".
   - *generateNewDESKey* that generates a random number between 6-10 and pass it to the *getRandomString* to generate a random key for DES implementation, storing it into *generatedDESKey.txt* in the *ThirdParty* folder.
   - *getKey* that reads and returns the generated new key for the current message from *generatedDESKey.txt* in the *ThirdParty* folder.
   - *encrypt* that reads the message in one given file and encrypt the message within using DES implementation and *getKey*, store the encrypted message in another file path passed in.
   - *decrypt* that reads the encrytion in one given file and decrypt the message within using DES implementation and *getKey*, store the decrypted message in another file path passed in.
2. ***FileManager*** includes:
   - *encrypt* that takes one given file and encrypt the file using DES in *javax.crypto.Cipher*, and store the encrypted file in another file path passed in.
   - *decrypt* that takes one given file and decrypt the encrypted file using DES in *javax.crypto.Cipher*, and store the decrypted file in another file path passed in.
   - *FileManager*, the constructor the automatically generates a DES secret key with *javax.crypto.spec.SecretKeySpec*.

3. ***User Interfaces***: The class ***Sender*** is the User Interface for Sender side, and it creates Receiver UI when the program is started. The class ***Receiver*** is the User Interface for Receiver side, both of them utilizes the ***Layout*** in the *UserInterface* package developed in project 1, which is a vertical flow layout for implementation.

4. ***Information folders***: The folder ***Sender*** stores a *senderTextMessage.txt* that contains a temporary message the sender sends. The folder ***Receiver*** contains *senderMessageDecrypted.txt* that stores a temporary decrypted message of the sender, the transmitted files will also be stored in this folder. ***ThridParty*** stores the temporary generated DES key for one message in *generatedDESKey.txt* and stores the encrypted temporary sender's message in *senderMessageEncrypted.txt*.

3 User Interface Design

The design of User Interfaces are as follows:

1. The user can the *DES.java*, and two windows will pop up as displayed below.

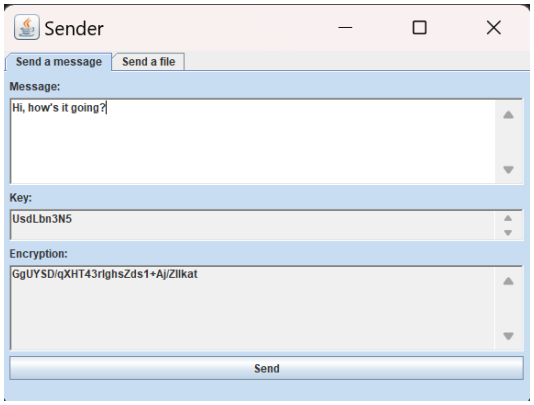

Figure 3: Sender (Send a message)

Figure 4: Receiver

2. The default is the "Send a message" page where the user can type a message in the "Message:" box and click on the "Send" button. On the sender's side, a key will be generated and displayed on the "key" box and the DES encryption of the message using the generated key will be displayed on the "Encryption:" box. On the receiver's side. The encryption, key and message will also be displayed on the corresponding boxes.

3. The user can also click on the "Send a file" to change the page to send a file. On the new page, the user can click on the "Select a file" button to select and open a file. On the sender's side, the file path, file name will be displayed on the "File information:" box.

4. After pressing the "Send" button, the file will be transfered to the receiver's side and the received file name will be displayed on the "Encryption:" box. As displayed in figure 2, the recieved file will be stored on the "Receiver" folder in the local file path as this application is just a demo. Moreover, if the user clicks the "Send" button without selecting a file, a window will pop up to ask the user to select a file first.

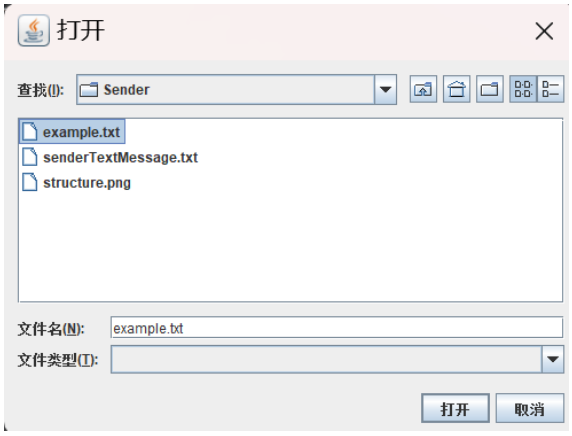

Figure 5: Select a file pop up

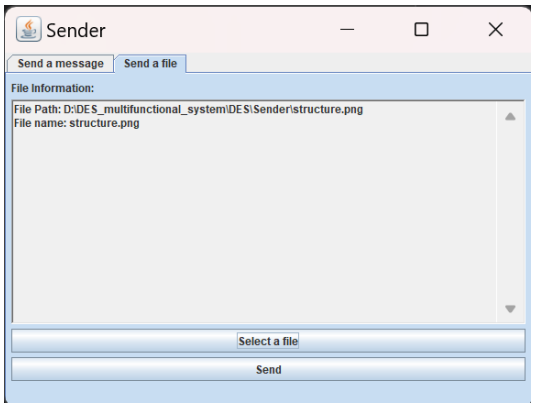

Figure 6: Sender (Send a file)

Figure 7: Receiver (file)

Figure 8: Please select a file pop up