Lab 2: Implementation and Application of DES

Although DES has been proved to be insecure and obsolete, it is still a good material for study and research since the DES algorithm exploits the Feistel block cipher structure that many modern symmetric ciphers are based on. Its enhanced version 3-DES is still widely used by industry and government. In this lab, we implement the basic version, DES.

Task

This is an **individual** work. Firstly, you need to generate a key and store it in a file (e.g., txt file). Then a simple chat program (socket programing) should be created to exchange messages between a Server ***S*** and a Client ***C***. Once the connection between ***S*** and ***C*** is successfully established, we can start exchanging messages. The client ***C*** uses DES algorithm to encrypt a message and sends the corresponding ciphertext to the server ***S***. The server ***S*** decrypts the received ciphertext to obtain the plaintext. Then the server ***S*** encrypts a new message and sends it back to the client ***C***. The client ***C*** receives the message from ***S*** and decrypts the received ciphertext to get the new message.

Steps

1. ***S*** and ***C*** should share a symmetric key for DES. Suppose ***C*** generates the key for DES and dumps the key to a file. Since we run the server and the client on the same computer here, we simply assume the server ***S*** has access to the key file.
2. ***S*** and ***C*** set up a simple chat program. (Socket)
3. ***S*** and ***C*** both load DES key from the key file on startup of their programs.
4. ***S*** and ***C*** exchange messages. The messages should be entered by you through keyboard in the console, **NOT** by hardcoding. All messages are encrypted before being sent.
5. The programs on both sides should display *the shared key*, *the plaintext message* to be sent, *the ciphertext* after encryption, *the received ciphertext*, and *the plaintext after decryption.*

Languages and Libraries

No restriction. Pick up whatever you are familiar with.

Submission

Submit the following files in a **zip** onto Canvas.

1. All of your source code files for this lab. Code comments will be helpful for the TA to understand your code.
2. A README file including information:
   a) Which language and external libraries you use.
   b) Which IDE you use.
   c) Detailed steps about how to run your code to finally obtain the required outputs. TA will check and run your code based on this instruction. If your code cannot be successfully executed, points will be deducted. **This is the most important part used to evaluate your work, so please be as specific as you can.**
3. A lab report including screenshots of each step of the entire code execution procedure, **as clear and specific as you can**. Alternatively, you can choose to use screen recording to record the entire code execution procedure and submit the recorded file.

Screenshots example

1. Connection established:

   Server side display:

```
Server is running...
Accept new connection from 127.0.0.1:53724...
```

2. Client types a message and sends message to server:

   Client side display:

```
Type message:

This is client.
******************
key is: imthekey
Sent plaintext is: This is client.
Sent ciphertext is: IW5cSjM5upYeWL1ST0ulVQ==
******************
```

   Server side display:

```
******************
received ciphertext is: IW5cSjM5upYeWL1ST0ulVQ==
received plaintext is: This is client.
******************
```

3. Server types a new message and sends back to client:

   Server side display:

```
Hi, this is server.
******************
key is: imthekey
Sent plaintext is: Hi, this is server.
Sent ciphertext is: skHlwD/LIiUrEqFAtsERGAq5CAklGEWX
******************
```

   Client side display:

```
******************
received ciphertext is: skHlwD/LIiUrEqFAtsERGAq5CAklGEWX
received plaintext is: Hi, this is server.
******************
```

Note that, for each single communication, you need to make sure the decrypted message on the receiver side should be the same as the plaintext that the sender sent. Otherwise, there must be something wrong with your code.

Submission Due: 11am, February 9, 2021.