INTI
LAUREATE INTERNATIONAL UNIVERSITIES

UNIVERSITY OF WOLLONGONG AUSTRALIA

FACULTY OF INFORMATICS

COURSEWORK COVERSHEET

SUBJECT'S INFORMATION:			
Subject:	CSCI361 Cryptography and Secure Applications		
Session:	July 2019		
Programme / Section:	BCS (S1, D1)		
Lecturer:	**Mohamad Faizal Alias**		
Coursework Type *(tick appropriate box)*	❑ Individual Assignment		
Coursework Title:	Assignment 4	Coursework Percentage:	10%
Hand-out Date:	Week 12	Received By : (signature)	
Due Date:	Week 16	Received Date :	
STUDENT'S INFORMATION:			
Student's Name & ID:	**TEH WIN SAM** **J16025133**		
Contact Number / Email:	**ME@TEHWINSAM.COM** **0133696298**		

STUDENT'S DECLARATION

By signing this, I / We declare that:

1. This assignment meets all the requirements for the subject as detailed in the relevant Subject Outline, which I/ we have read.
2. It is my / our own work and I / we did not collaborate with or copy from others.
3. I / we have read and understand my responsibilities under the University of Wollongong's policy on plagiarism.
4. I / we have not plagiarised from published work (including the internet). Where I have used the work from others, I / we have referenced it in the text and provided a reference list at the end of the assignment.

I am / we are aware that late submission without an authorised extension from the subject co-ordinator may incur a penalty. *(See your subject outline for further information).*			
Name & Signature:			

----------✂ ----------✂ ----------✂ ----------✂ ----------✂ ----------✂ --

COURSEWORK SUBMISSION RECEIPT

Subject:	CSCI361 Cryptography and Secure Application	Session:	July 2019
Programme / Section:	BCS	Lecturer:	**Mohamad Faizal Alias**
Coursework Type: (Tick appropriate box)	❑ Individual Assignment		
Coursework Title:	Assignment 4	Coursework Percentage:	10%
Hand-out Date:	Week 12	Received By: (Signature)	
Due date:	Week 16	Received Date:	
STUDENT'S INFORMATION:			
Student's Name & ID:			
Contact Number / Email:			

Assessment Criteria		**Total Marks**	**Given Marks**
1.	Client-Server Execution	10	
2.	Client - Key Generation coding **RSA (PUa & PRa)**	10	
3.	Server Verify **PUa** with Hash of **PUa** – received from Client	10	
4.	Server IDEA Key Gen (**Ks**) & MD-5 Hash for **Ks + E(PUa, Ks)**	20	
5.	Client decrypt **D(PRa, Ks)** and Verify Hash of Ks	10	
6.	Encrypted data communication process – **3DES with CBC Mode**	20	
7.	Overall report & Presentation	20	
		100	
		Penalty	
Marked by: _____ Date: _____		**Final Mark (10 %)**	
Lecturer's Comments			

Penalty for late submission:
1 day – minus 20% of total mark awarded 2 days – minus 50% of total mark awarded 3 days – 0 mark for this piece of coursework

University of Wollongong

CSCI361: Cryptography and Secure Applications

July 2019 Session

Individual Assignment 4 (10 %)

Tasks:

- Extend and implement of the **RSA** PKC scheme in station-to-station communication
- Using Hashing for integrity of message, that is **MD-5**
- Produce simple Key Transport protocol
- **3DES** encryption with 2 Keys a.k.a. 3DES-EDE2
- Mode of Block Cipher is **CBC Mode**

Specifications:

The aim of this assignment is to create a program that allows two different stations to communicate by initiating the following protocols:

Some Guidelines before you start:

1. Those who planned to use VM or VirtualBox, make sure that you've installed VM/VirtualBox and configure the Network and set the configuration to use Bridge mode
2. After setting up VM/VirtualBox with Bridge mode only install your Linux or other OSes that you preferred.
3. Each time you test the server and the client program, the server need to open a port (port number). Any error during testing your OS might not release the port number and/or the port currently having buffer of previous data. This might lead to error on your second-time execution of your program. It is suggested that each time you execute server, change the port number and let the client program use the new port number opened.
4. Developed both server and client code in separate folders. This to ensure any local copy of reference file (if so exists) won't give you a misleading error or/and success of the code execution.
5. Use any C++ sample of socket programming and execute them first to ensure that the sample is working on real network environment. It is NOT suggested to use WinSock since it might bound to restriction in Windows environment.
6. You can use two computers and connect them via Ethernet cable (CAT5 cable) directly to simulate a network.
7. You may be interested to venture into multithreading
8. Observed carefully the protocol given below:

Handshake Process:

 Client Program	Preferable your program is proven to be running on the real network NOT only on Localhost with VMWare 	 Server Program
		1. Start a Host (server) – Initiate by entering the port no. for client to connect
2. Execute client – enters the IP and Port number of the Server 3. Client program: starts with generating key pairs; PU_A-Public Key and PR_A-Private Key using **RSA** 4. PU_A will be send over to the server, together with the Hash of PU_A using **MD-5** for integrity checking. PR_A is kept by Client for later Decryption.	PUa || H(PUa) || Flag to connect (optional) 	5. Upon receiving PU_A and H(PU_A), Verify them using **MD-5**. 6. Server generates a session key **K_s1** (**Key size of 8 Bytes)** for the purpose of two keys **3DES encryption** 7. Next, the Server prepares $K_s = K_s1 + K_s2$ K_s1 is the 8 bytes key while K_s2 is the **reverse of K_s1**
	E(PUa, Ks1) || H(Ks1) || Flag for acknowledgement (optional) 	8. Server then sends over to Client, encrypted **K_s1** (**8 Bytes**) using PU_A that is E(PU_A, **K_s1**) and H(**K_s1**)
9. Upon Receiving of E(PU_A, **K_s1**) and H(**K_s1**); Client Decrypt the message using PR_A , get to know **K_s1** (**3DES Key size 8 Bytes)**. 10. Then it verifies the integrity of **K_s1** by checking H(**K_s1**) – Hashing via **MD-5**		

Note:

- Prepare your program so that it will request the user to enter appropriate settings such as IP number and port number.
- Connection is using socket programming either using UDP or TCP.
- You may use all functionalities available in Crypto++ library
- Flags are optional depending on your technique to control the handshake process.
- IV for CBC mode can be hardcoded in both of your Client and Server code
- Reference: https://www.cryptopp.com/wiki/TripleDES

Data Communication Process:

 Client Program		 Server Program
11. Upon completion of Handshake process above, Client is now use **K_s1** to generate **K_s2** by reversing **K_s1** to produce the complete 3DES Key (**$K_s1 + K_s2 = K_s$**) to encrypt its data communication 12. Client program: Encrypt the message **Mc** using **3DES** encryption with the agreed session **Ks1+ K_s2** (**3DES Key size**) 13. Message **Mc** enters **3DES** encryption using **CBC Mode** 17. Upon receiving message **Ms** from server, client can decrypt it using the session **K_s** 18. Communication continues until '**Quit'** signal initiated	E(Ks, M) || Flag – from Client E(Ks, M)|| Flag – from Server	13. Server then Decrypt the message **Mc** from client using **$K_s1 + K_s2 = K_s$** 14. Server program: Encrypt the message **Ms** using **3DES** encryption with the agreed session **K_s** (**3DES**) **15.** Message **Ms** enters **3DES** encryption using **CBC Mode** 16. Communication continues until '**Quit'** signal initiated

Note:

- The working of message sending and receiving works something like a chat program.
- All the above decisions and reusable library must be reported and reference accordingly
- Data communication process between A and B can be done either through VMware or a more appropriate approach is over the real network (you can use 2 computers for this purpose during presentation)
- Flags are optional – depending on how you control the data communication

Reporting

You are to include in your report all the following (but not bound to only these requirements):

1. Setting-up of the Stations involved simulation requirements etc.
2. All cryptosystem and Hashing strategies implemented
3. Discussion on the execution (steps) of your program
4. program explanation (on all methods used), overall program structure, data input/output, analysis results
5. Other requirements deem important

Submission

Your source code should be submitted together with the report.pdf file in one folder. ZIP the folder and named it <your-name-CSC361-Assign4>.zip and submit this ZIP file to Moodle Submission link provided. Remember to put your name and student number in all source codes (comments header).

Provide readme.txt file to guide me on your library used, execution, setting up of IP number and port examples; and/or other deemed important. Make sure the folder also has error free compiled version of your program(s).

Assignments must be submitted electronically via Moodle submission link.

Presentation

A short presentation of your working program is required. Presentation slot will be announced by your lecturer during your face-to-face class session.

Plagiarism

A plagiarised assignment will receive a zero mark (and penalised according to the university rules). Plagiarism detection software will be used.

Table of Contents

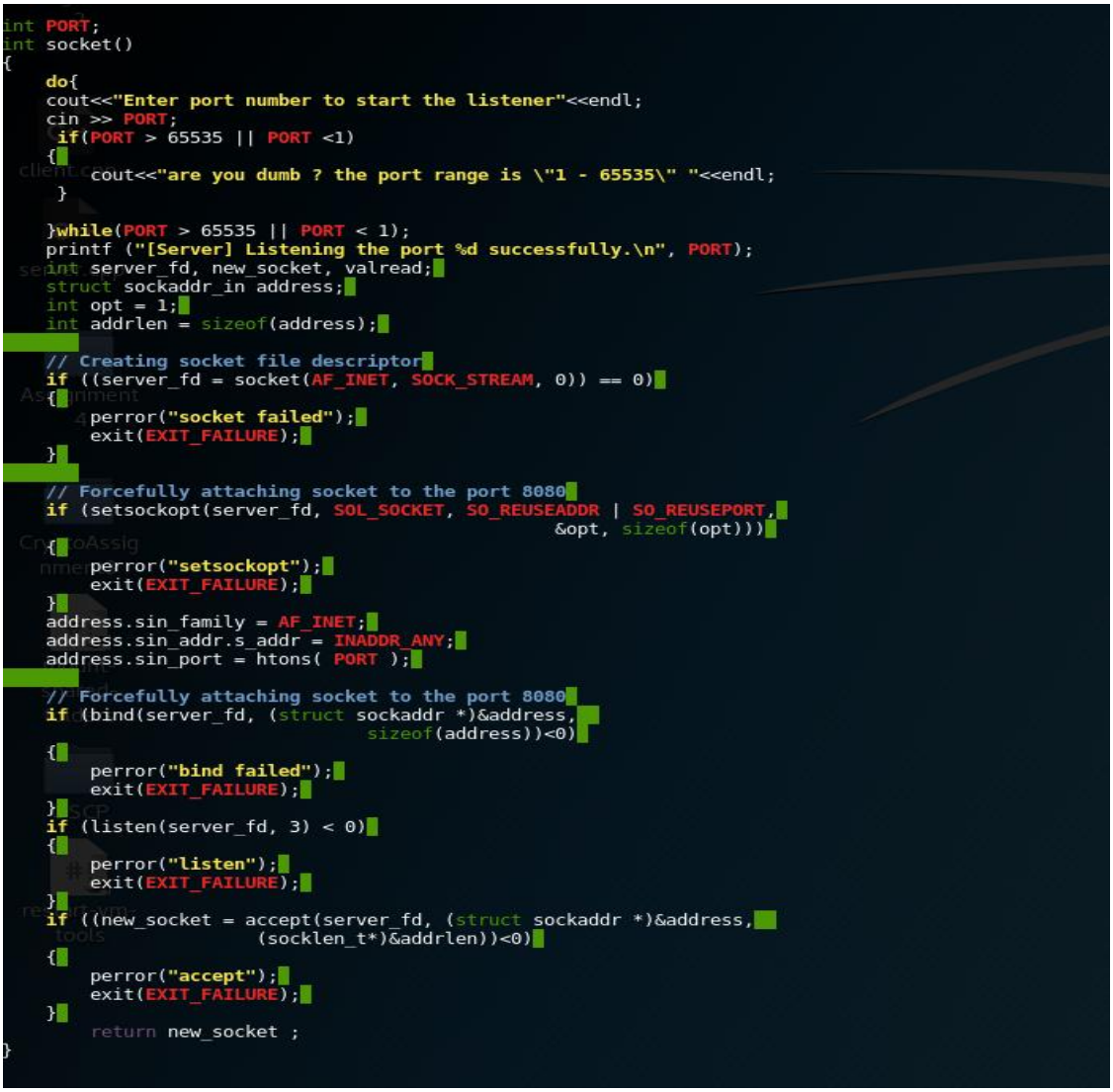
1. Start a Host (server) – Initiate by entering the port no. for client to connect

On Server's code, the first line in main() is to assign datatype int and get the value from socket().

```
int new_socket=socket()
int valread;
```

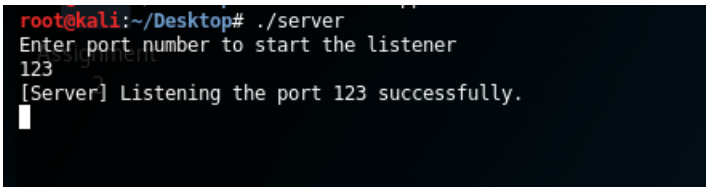
Socket()

Prompt user to input "port number " to start the listener, and when it successfully listening on the port it state "Listening the port 123 successfully", assume the user input "123" as port number



```
int PORT;
int socket()
{
    do{
    cout<<"Enter port number to start the listener"<<endl;
    cin >> PORT;
     if(PORT > 65535 || PORT <1)
    {
        cout<<"are you dumb ? the port range is \"1 - 65535\" "<<endl;
     }

    }while(PORT > 65535 || PORT < 1);
    printf ("[Server] Listening the port %d successfully.\n", PORT);
    int server_fd, new_socket, valread;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);

    // Creating socket file descriptor
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    // Forcefully attaching socket to the port 8080
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
                                                  &opt, sizeof(opt)))
    {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons( PORT );

    // Forcefully attaching socket to the port 8080
    if (bind(server_fd, (struct sockaddr *)&address,
                                 sizeof(address))<0)
    {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
    if (listen(server_fd, 3) < 0)
    {
        perror("listen");
        exit(EXIT_FAILURE);
    }
    if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                        (socklen_t*)&addrlen))<0)
    {
        perror("accept");
        exit(EXIT_FAILURE);
    }
        return new_socket ;
}
```

Output of listening port



```
root@kali:~/Desktop# ./server
Enter port number to start the listener
123
[Server] Listening the port 123 successfully.
```

2. Execute client – enters the IP and Port number of the Server

From the client's main()first line of code I declare an int datatype and will grab the value from socket()

```
{
    int sock=socket(),valread;
    char buffer[1024] = {0};
```

Socket()

```
int socket()
{
    cout<<"Enter Server IP ADDRESS"<<endl;
    string ip;
    cin>>ip;
    int PORT;
    do{
    cout<<"Enter port number"<<endl;
    cin>>PORT;
    if(PORT > 65535 || PORT <1)
    {
        cout<<"are you dumb ? the port range is \"0 - 65535\" "<<endl;
     }
    }while(PORT > 65535 || PORT < 1);
    int sock = 0, valread;
    struct sockaddr_in serv_addr;
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("\n Socket creation error \n");
        exit(0);
        return -1;
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    // Convert IPv4 and IPv6 addresses from text to binary form
    if(inet_pton(AF_INET, ip.c_str(), &serv_addr.sin_addr)<=0)
    {
        printf("\nInvalid address/ Address not supported \n");
        exit(0);
        return -1;
    }

    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
    {
        printf("\nConnection Failed \n");
        exit(0);
        return -1;
    }
        return sock;
}
```

This is the socket function will start to connect to server by asking user to input the "Server IP" and "Server port". "If connection is establish then it will return sock value" else it state connection failed.

Output of connecting to server

```
root@kali:~/Desktop# ./client
Enter Server IP ADDRESS
127.0.0.1
Enter port number
123
```

3. Client program: starts with generating key pairs; PU_A-Public Key and PR_A-Private Key using RSA

keyGen()

Will perform key generation with rng (1024 bit key) and store into "assignment4publickey.txt" and "assignment4privatekey.txt"

```
void keyGen()
{
        AutoSeededRandomPool rng;
        InvertibleRSAFunction privkey;
        privkey.Initialize(rng, 1024);

    // Generate Private Key
    RSA::PrivateKey privateKey;
    privateKey.GenerateRandomWithKeySize(rng, 1024);
    // Generate Public Key
    RSA::PublicKey publicKey;
    publicKey.AssignFrom(privateKey);
    SaveHexPublicKey("assignment4publickey.txt", publicKey);
    SaveHexPrivateKey("assignment4privatekey.txt", privateKey);
}
```

IN HEX format because of following function

```
//code from stackoverflow
//https://stackoverflow.com/questions/29050575/how-would-i-load-a-private-public-key-from-a-string-byte-array-or-any-other
void Save(const string& filename, const BufferedTransformation& bt)
{
    FileSink file(filename.c_str());
    bt.CopyTo(file);
    file.MessageEnd();
}

void SaveHex(const string& filename, const BufferedTransformation& bt)
{
    HexEncoder encoder;
    bt.CopyTo(encoder);
    encoder.MessageEnd();
    Save(filename, encoder);
}

void SaveHexPrivateKey(const string& filename, const PrivateKey& key)
{
    ByteQueue queue;
    key.Save(queue);
    SaveHex(filename, queue);
}

void SaveHexPublicKey(const string& filename, const PublicKey& key)
{
    ByteQueue queue;
    key.Save(queue);
    SaveHex(filename, queue);
}
```

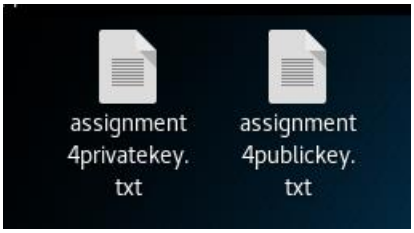
In main() it perform a "display text of RSA Key is generating" then only proceed to keygen() function.

```
cout<<FRED(BOLD("[System] RSA Key is generating"))<<endl;
keyGen();
string privatekey=grabprivatekey("assignment4privatekey.txt");
string publickey=grabfilecontent("assignment4publickey.txt");
```

And grab the content from "these 2 following files" and "using 2 different functions" into 2 variable "privatekey" and "publickey"

```
string grabfilecontent(string filename)
{
 string inputdata,totaldata;
 ifstream file (filename);
  if (file.is_open())
  {
int counter=0;
   while(getline (file,inputdata))
     {
                totaldata=totaldata+inputdata+"\n";
     }
    file.close();
        return totaldata;
  }
}

string grabprivatekey(string filename)
{
 string inputdata,totaldata;
 ifstream file (filename);
  if (file.is_open())
  {
        getline (file,inputdata);
            file.close();
        return inputdata;
  }
}
```

Output “PublicKey and PrivateKey” File Created



4. PU_A will be send over to the server, together with the Hash of PU_A using MD-5 for integrity checking. PR_A is kept by Client for later Decryption.

```
send_recv(sock, publickey, "Public Key have sent");
```

First, it will send “publickey variable data” over the network to Server in “plaintext(HEXA FORMAT)”.

```
cout<<FRED(BOLD("----------PUBLIC KEY AND HASH THAT SEND OVER NETWORK----------"))<<endl;
cout<<FRED(BOLD("Value of Public Key"))<<endl;
cout<<publickey<<endl;
string publicmd5=md5string(grabfilecontent("assignment4publickey.txt"));
cout<<FRED(BOLD("[MD5]PUBLIC KEY : "));
cout<<publicmd5<<endl;

send_recv(sock, publicmd5, "Hash value \"MD5\" of public key sent");
cout<<FRED(BOLD("----------E O F----------"))<<endl;
```

After send over the network to Server, it will display the “value of publickey in HEXA format”, and perform a hashing on the “publickey” using MD5 function *image below* and send another data (Digest checksum) to SERVER to verify the “publickey” is not tampered.

```
string md5string(string haha)
{
#define CRYPTOPP_ENABLE_NAMESPACE_WEAK 1

byte digest[ CryptoPP::Weak::MD5::DIGESTSIZE ];
std::string message = haha;

CryptoPP::Weak::MD5 hash;
hash.CalculateDigest( digest, (const byte*)message.c_str(), message.length() );

CryptoPP::HexEncoder encoder;
std::string output;

encoder.Attach( new CryptoPP::StringSink( output ) );
encoder.Put( digest, sizeof(digest) );
encoder.MessageEnd();

return output;
}
```

Output Content of PublicKey and MD5 Digest that send over network

```
[ Msg from Server ] --> [ Client ] : Received Wink > .. <
----------PUBLIC KEY AND HASH THAT SEND OVER NETWORK----------
Value of Public Key
30819D300D06092A864886F70D010101050003818B00308187028180B2E8ED16DECC37E8382EADAC91A5F5E2934C0518A3C63F9E094AD56EC062249D2AD150C658B6D65A670C3C72B6567A6C660D589E7B
553245E047E03F87FA6AC3DE493C39E12F96F5A5E07F651454F23DCC399262858610EA07923B61596EC7E02EA15F8190CD77D042577493279DB17D3DD4F26DB36E58E06C13208F89AFAB91020111

[MD5]PUBLIC KEY : CF520FF4B666A5D47F65DB81CBC8DB77
[ Successfully send to Server ] Hash value "MD5" of public key sent
Waiting/Receiving Message ...
[ Msg from Server ] --> [ Client ] : Received Wink > .. <
----------E O F----------
```

5. Upon receiving PU_A and $H(PU_A)$, Verify them using MD-5.

In the server side's main() function, it will grab the content that send from client into to publickey and hashvalue using *send_recv() function* since **send_recv is returning "string value"**

```
string send_recv(int new_socket, string message, string comments)
{
    int valread;
    char buffer[1024] = {0};
        string compare;
        valread = read( new_socket , buffer, 1024);
        cout<<"[ Msg from Client ] --> [ Server ] : "<<buffer<<endl;
        send(new_socket , message.c_str() , strlen(message.c_str())+1 , 0 );
        cout<<"[ Details ] "<<comments<<endl;
return buffer;
}
```

```
string publickey=send_recv(new_socket, hello, "Public key from Client");//send received wink
SaveContent(publickey,"received_publickey.txt");
string hashvalue=send_recv(new_socket, hello, "Hash value from Client"); // send received wink
string contentofpublickeytoverify=grabfilecontent("received_publickey.txt");
verify(hashvalue,md5string(contentofpublickeytoverify));
```

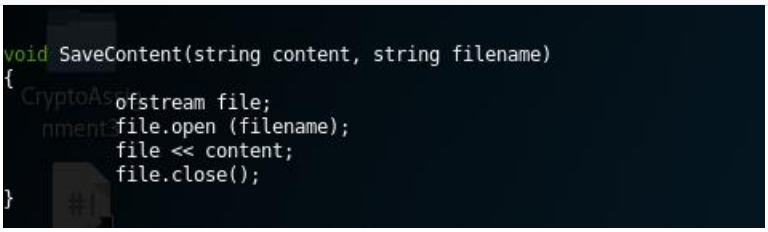
Do notice the second and third parameter, second parameter is just a "variable from server say to client that 'noted I haved received my message' and second parameter 'is just to display' what kind of content that server is receiving ".

```
   string publickey=send_recv(new_socket, hello, "Public key from
Client");//send received wink
```

```
   string hashvalue=send_recv(new_socket, hello, "Hash value from Client");
// send received wink
```

And according to main() will save the the publickey content into a filename called "received_public.key"

```
   SaveContent(publickey,"received_publickey.txt");
```



```
void SaveContent(string content, string filename)
{
    ofstream file;
    file.open (filename);
    file << content;
    file.close();
}
```

And we have to verify the "publickey" content with "digest" value that sent from client.

```
void verify(string a, string b)
{

int result = strcmp(a.c_str(), b.c_str());
cout<<"Verifying integrity of file"<<endl;
if(result==0)
{
        cout<<BOLD(FRED("Matched"))<<endl;
}
else
{
        cout<<a<<endl;
        cout<<b<<endl;
        cout<<BOLD(FRED("NOT MATCH!"))<<endl;
        exit(0);
}
}
```

If the values of “PUBLICKEY’s digest” and “DIGEST received from Client” not match then it will “Display not match” and quit the program else it will continue proceed the program.

Output of integrity checking

```
Verifying integrity of file
Matched
```

6. Server generates a session key K_s1 (Key size of 8 Bytes) for the purpose of two keys 3DES encryption

Complete function 'string generateSessionKey(int sock,string publickey)' at section 7

```
        AutoSeededRandomPool prng;
        InvertibleRSAFunction parameters;
        RSA::PublicKey publicKey(parameters);
        parameters.GenerateRandomWithKeySize(prng,1024);
        int keys=8;
        SecByteBlock key(keys);
        prng.GenerateBlock(key, key.size());
        //Convert key from bytes to string
        string stringKey,temporary;
        ArraySource (key, sizeof(key), true, new StringSink(stringKey));
        string encodestringKey;
        StringSource encodekey(stringKey, true, new HexEncoder(
        new StringSink(temporary)));
        encodestringKey=temporary.substr(0,16);
        cout<<BOLD(FRED("[--------------------------------SESSION KEY IS
GENERATING-----------------------------]"))<<endl;
        cout<<"EncodedSessionKey Ks1 : "<<encodestringKey<<endl;
.
.
.
StringSource decodekey(publickey,true,new HexDecoder( new
StringSink(decodedpubkey)));
        StringSource pubKeySS(decodedpubkey,true);
        publicKey.Load(pubKeySS);

        string encryptedSessionKey;
        RSAES_OAEP_SHA_Encryptor e(publicKey);
        StringSource encryptboth(encodestringKey, true, new
PK_EncryptorFilter(prng,e,(new HexEncoder(new
StringSink(encryptedSessionKey)))));
        cout<<BOLD(FRED("[--------------------------------SESSION KEY
GENERATE COMPLETED----------------------------]"))<<endl;
        cout<<BOLD(FRED("[-------------------------PERFORM ENCRYPTION ON
SESSION KEY USING PUBLIC KEY -----------------------------]"))<<endl;
        cout<<"[ENCRYPTED]"<<encryptedSessionKey<<endl;

        return encryptedSessionKey;
```

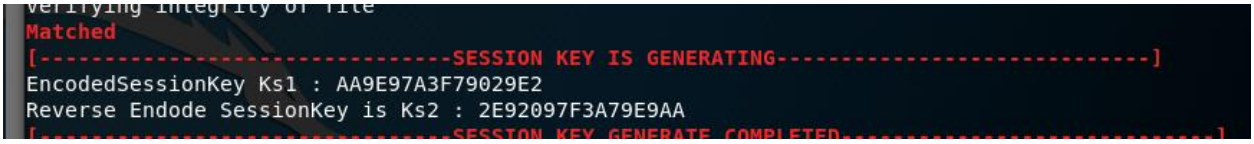
Output of Ks1 EncodedSessionKey using HexEncoder

7. Next, the Server prepares $K_s = K_s1 + K_s2$ | K_s1 is the 8 bytes key while K_s2 is the **reverse of K_s1**

```
string generateSessionKey(int sock,string publickey)
{
      AutoSeededRandomPool prng;
      InvertibleRSAFunction parameters;
      RSA::PublicKey publicKey(parameters);
      parameters.GenerateRandomWithKeySize(prng,1024);
      int keys=8;
      SecByteBlock key(keys);
      prng.GenerateBlock(key, key.size());

      //Convert key from bytes to string
      string stringKey,temporary;
      ArraySource (key, sizeof(key), true, new StringSink(stringKey));
      string encodestringKey;
      StringSource encodekey(stringKey, true, new HexEncoder(
      new StringSink(temporary)));
      encodestringKey=temporary.substr(0,16);
      cout<<BOLD(FRED("[----------SESSION KEY IS GENERATING----------]"))<<endl;
      cout<<"EncodedSessionKey Ks1 : "<<encodestringKey<<endl;
      thirdpairkeyfordes=encodestringKey;
      string  tmp;
      string reverse_sessionkey;

      for(int i=0; i <= encodestringKey.length();i++)
      {
            tmp[i]=encodestringKey[encodestringKey.length()-i];
             reverse_sessionkey= reverse_sessionkey+tmp[i];
      }
      firstkey=encodestringKey;
      cout<<"Reverse Endode SessionKey is Ks2 : "<<reverse_sessionkey<<endl;;
      secondkey=reverse_sessionkey;
      hashvaluesessionkey=encodestringKey;
      string decodedpubkey;
      StringSource decodekey(publickey,true,new HexDecoder( new StringSink(decodedpubkey)));
      StringSource pubKeySS(decodedpubkey,true);
      publicKey.Load(pubKeySS);

      string encryptedSessionKey;
      RSAES_OAEP_SHA_Encryptor e(publicKey);
      StringSource encryptboth(encodestringKey, true, new PK_EncryptorFilter(prng,e,(new HexEncoder(new StringSink(encryptedSessionKey)))));
      cout<<BOLD(FRED("[----------SESSION KEY GENERATE COMPLETED----------]"))<<endl;
      cout<<BOLD(FRED("[----------PERFORM ENCRYPTION ON  SESSION KEY USING PUBLIC KEY ----------]"))<<endl;
      cout<<"[ENCRYPTED]"<<encryptedSessionKey<<endl;

      return encryptedSessionKey;
}
```

Which will return the encrypted value of “e(Ks1)” which is using publickey to encrypt .

Output of Ks2 EncodedSessionKey using HexEncoder



8. Server then sends over to Client, encrypted K_s1 (8 Bytes) using PU_A that is E(PU_A, K_s1) and H(K_s1)

In main(), it will generateSessionKey and return the **encrypted session key** into variable **"sakeofreturn"** and it will send it over to network using "send_recv()" function and inside the function will display "Server --> Client | Encrypted Session Key" on Server's terminal

```
  sakeofreturn=generateSessionKey(new_socket,publickey);
   send_recv(new_socket, sakeofreturn, "Server --> Client | Encrypted Session
Key");
```

Next it will perform another a MD5 hash on the **"ENCODEDSESSIONKEY"** NOT **ENCRYPTEDSESSIONKEY"** and store into variable **md5sessionencode** send over the network using "send_recv()" function and inside the function will display "Server --> Client | MD5 Session Key" on Server's terminal

```
   string md5sessionencode=md5string(hashvaluesessionkey);
   cout<<BOLD(FRED("[----------------------------------END OF FILE------------
------------------------------]\n\n"))<<endl;
   cout<<BOLD(FRED("[-------------------------GENERATING HASH VALUE OF
ENCRYPTED SESSION KEY FROM CLIENT ----------------------------]"))<<endl;
   cout<<FRED(BOLD("MD5 of the PUBLICKEY + SESSION KEY : "));
   cout<<"\""<<md5sessionencode<<"\" "<<endl;
   sakeofreturn=send_recv(new_socket, md5sessionencode, "Server --> Client |
MD5 Session Key");
```

Output of ENCRYPTED SESSION and Digest from Server Perspective

9. Upon Receiving of E(PUA, Ks1) and H(Ks1); Client Decrypt the message using PRA , get to know Ks1 (3DES Key size 8 Bytes).

In client's main()

```
cout<<FRED(BOLD("-----------------------------------------------RECEIVED
ENCRYPTED SESSION USING PUBLIC KEY AND IT HASH VALUE FROM SERVER------------
------------------------------------------------------------"))<<endl;

string encryptedsession=send_recv(sock, recieve, "Encrypted Session Key from
Server"); // received encrypted session key from server

send_recv(sock, recieve, "Hash value \"MD5\" of Session Key "); // received
hash session key from server

cout<<FRED(BOLD("-----------------------------------------------EOF---------
------------------------------------------------------------"))<<endl;
```

After received "*encryptedsession in global variable*" and "*hashencodedsession in global variable*" that sent from Server , it will perform decryption on variable "*encryptedsession*" in DecryptSession() function.

```
void DecryptSession(string session,string privKey)
{
string decodedEncHexEnSeshKey;
StringSource ss(session,true,new HexDecoder(new StringSink(decodedEncHexEnSeshKey)));

AutoSeededRandomPool rng;
InvertibleRSAFunction parameters;
parameters.GenerateRandomWithKeySize(rng,1024);
RSA::PrivateKey privateKey(parameters);
string decodedPrivKey;

StringSource ss2(privKey,true,(new HexDecoder( new StringSink(decodedPrivKey))));       //decode the privkey from hex to symbol stuff
StringSource PrivKeySS(decodedPrivKey,true);           //load it into bytes
privateKey.Load(PrivKeySS);            //load the private key

RSAES_OAEP_SHA_Decryptor d(privateKey);
string hexEnSeshkey;
StringSource ss3(session ,true,(new HexDecoder (new PK_DecryptorFilter(rng, d, (new StringSink(hexEnSeshkey))))));
cout<<"-----------------------------------------------Decryption is in progress .. .. .. . . .-------------------------------------"<<endl;
cout<<BOLD(FRED("[ *Session Key found* ] "));
cout<<hexEnSeshkey<<" | MD5 : "<<md5string(hexEnSeshkey)<<endl;
cout<<"-----------------------------------------------Process of decryption is completed-------------------------------------"<<endl;

firstkey=hexEnSeshkey;
encodedsessionkey=hexEnSeshkey;
        string  tmp;
       string reverse_sessionkey;
       for(int i=0; i <= encodedsessionkey.length();i++)
        {
             tmp[i]=encodedsessionkey[encodedsessionkey.length()-i];
              reverse_sessionkey= reverse_sessionkey+tmp[i];
        }
       cout<<"Reverse Endode SessionKey is Ks2 : "<<reverse_sessionkey<<endl;;
secondkey=reverse_sessionkey;
}
```

It will use the privKey variable and perform HexDecoder and store into decodedPrivKey and load as privateKey as bytes and use the RSAES_OAEP_SHA_Decryptor load the privateKey and perform "HexDecoder and Decryption using PK_DecryptorFilter" into variable hexEnSeshkey. After finishing the decryption process we generate a checksum MD5 using value that store in "hexEnSeshkey". So that we can perform the verification processs.

```
string send_recv(int socket, string message, string comments)
{
int valread;
char buffer[1024] = {0};
send(socket , message.c_str() , strlen(message.c_str())+1 , 0 );
cout<<"[ Successfully send to Server ] "<<comments<<endl;
cout<<"Waiting/Receiving Message ... "<<endl;
valread = read(socket , buffer, 1024);
cout<<"[ Msg from Server ] --> [ Client ] : ";
printf("%s\n",buffer );
hashencodedsession=buffer;
encryptedmsg=buffer;
return encryptedmsg;
}
```

And in send_recv() I do create a variable that store data into hashencodedsession, which mean in previous those publickey that send over the network will also store into hashencodedsession, but when it came to receiving “hash value on encodedsession” I will only use this variable.

Output Received EncryptedSession in HEX and Digest value from Server at Client terminal’s Perspective and perform decryption process.

```
----------RECEIVED ENCRYPTED SESSION USING PUBLIC KEY AND IT HASH VALUE FROM SERVER----------
[ Successfully send to Server ] Encrypted Session Key from Server
Waiting/Receiving Message ...
[ Msg from Server ] --> [ Client ] : CB3797D2A89643B519F5D0409FF11FFB37C53A814D290DE991048BA274B09CE7649F562A2AF69CF8C28AAB89BD6EA81BDFFCD384D70BA2A88DEAFE32C7D8E228D4DDF139BA3FFC1A3D9C4F4ED53C430166B6AB3E
1BD5CC4F15F086CF45B54B9BDFF682DEA1FA29A58DA0489F0D37087A97774CC6165C23FBCCFDF94379AC5D6B
[ Successfully send to Server ] Hash value "MD5" of Session Key
Waiting/Receiving Message ...
[ Msg from Server ] --> [ Client ] : B85866279821F66D410565BE7D0DDC06
----------EOF----------
----------Decryption is in progress .. .. .. . . .----------
[ *Session Key found* ] AA9E97A3F79029E2 | MD5 : B85866279821F66D410565BE7D0DDC06
----------Process of decryption is completed----------
```

10. Then it verifies the integrity of K_s1 by checking H(K_s1) – Hashing via MD-5

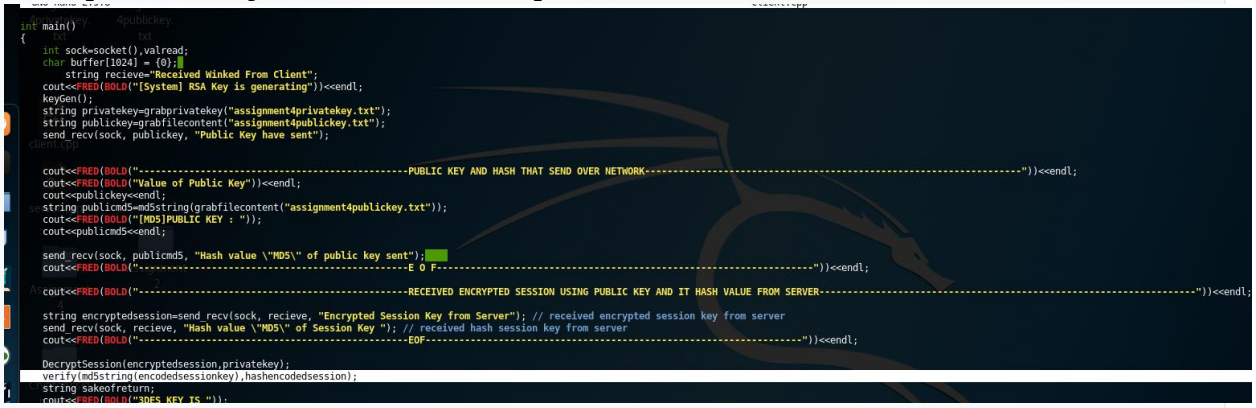
The value of encodedsessionkey and hashencodedsession is global variable.
which you can find on section 9 how I generate and assign the value into these variables.

After finished decryption on the encryptedsessionkey.

Now it will go into Verify() to verify both match, whether it matches or not .

Code in main()

```
verify(md5string(encodedsessionkey),hashencodedsession);
```




```
Md5string() function can be found in section 4
```

Verify()

```
void verify(string a, string b)
{

int result = strcmp(a.c_str(), b.c_str());
cout<<"Verifying integrity of file"<<endl;
if(result==0)
{
        cout<<BOLD(FRED("Matched"))<<endl;
}
else
{
        cout<<a<<endl;
        cout<<b<<endl;
        cout<<BOLD(FRED("NOT MATCH!"))<<endl;
        exit(0);
}
}
```

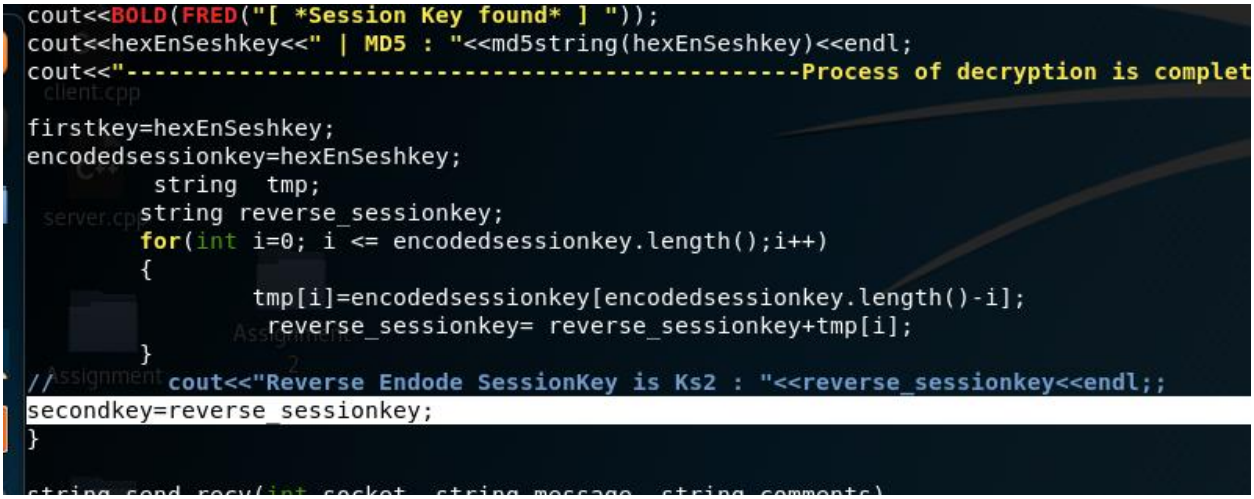
So if the SessionKey's intergrity is not match it will terminate the program. If the sessionkey's intergrity is match then it will continue it program.

Output Verifying intergrity of it files.

```
----------Process of decryption is completed----------
Verifying integrity of file
Matched
Reverse Endode SessionKey is Ks2 : E3924EC18EDC753F
```

11. Upon completion of Handshake process above, Client is now use K_s1 to generate K_s2 by reversing K_s1 to produce the complete 3DES Key ($K_s1 + K_s2 = K_s$) to encrypt its data communication

Back to the section 9, which I have already reverse it. It will store at secondkey, and secondkey is a globalvariable.



```
cout<<BOLD(FRED("[ *Session Key found* ] "));
cout<<hexEnSeshkey<<" | MD5 : "<<md5string(hexEnSeshkey)<<endl;
cout<<"-------------------------------------------------Process of decryption is complet
client.cpp

firstkey=hexEnSeshkey;
encodedsessionkey=hexEnSeshkey;
        string  tmp;
       string reverse_sessionkey;
       for(int i=0; i <= encodedsessionkey.length();i++)
       {
               tmp[i]=encodedsessionkey[encodedsessionkey.length()-i];
               reverse_sessionkey= reverse_sessionkey+tmp[i];
       }
//      cout<<"Reverse Endode SessionKey is Ks2 : "<<reverse_sessionkey<<endl;;
secondkey=reverse_sessionkey;
}
```

12. Client program: Encrypt the message Mc using 3DES encryption with the agreed session Ks1+ $K_{s}2$ (3DES Key size)
13. Message Mc enters 3DES encryption using CBC Mode

```
verify(md5string(encodedsessionkey),hashencodedsession);
cout<<"Reverse Endode SessionKey is Ks2 : "<<secondkey<<endl;;
string sakeofreturn;
cout<<FRED(BOLD("3DES KEY IS "));
cout<<"\""<<firstkey<<secondkey<<"\""<<endl;
cout<<FRED(BOLD("HANDSHAKE ESTABLISHED\n\n----------"))<<endl;
string tripledeskey=firstkey+secondkey;
bool loop=true;
cin.ignore();
do
{
tripleDES_encrypt(tripledeskey,sock);
tripledes_decrypttext(sock,tripledeskey,encryptedmsg);
}while(loop==true);
```

It will grab the “firstkey”(global variable) + “secondkey” (global variable) which can be found in section 9 and store combine into a variable called tripledeskey and pass into tripleDES_encrypt() and tripledes_decrypttest() as key to perform the decryption and encryption.

In the encryption process, I have a “hardcoded” iv bytes and uses DES_EDE2 to perform the encryption process.

First the key will perform HexDecoder and store as decodedkey.Next convert the decodedkey into variable key as SecBytesBlock type.

Next will prompt user to input the message to perform encryption and the text length cannot more than 500 length.

Next will create an CBC_MODE DES_EDE2 Encryption object and setkeywithiv and perform encryption process and store the output into variable cipher .

Next use HexEncoder to encode the encryptedtext(cipher) and store as encoded

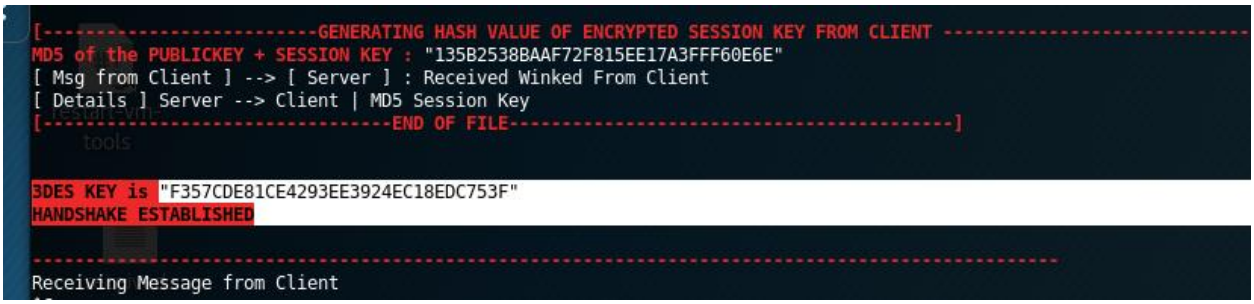
```
void tripleDES_encrypt(string keys,int sock)
{
AutoSeededRandomPool prng;
        string decodedkey;
     StringSource s(keys, true,(new HexDecoder(
                new StringSink(decodedkey))
        ) // StreamTransformationFilter
      ); // StringSource

SecByteBlock key((const byte*)decodedkey.data(), decodedkey.size());

const byte iv[] = {0x12,0x34,0x56,0x78,0x90,0xab,0xcd,0xef};

string plain;
string cipher, encoded, recovered;

/*********************************\
\*********************************/
do
{
cout<<"Enter message send to server"<<endl;
std::getline(std::cin, plain);
if(plain.size()>500)
{
cout<<BOLD(FRED("Message is exceed the length"))<<endl;
}
}while(plain.size()>500);
try
{

    CBC_Mode< DES_EDE2 >::Encryption e;
    e.SetKeyWithIV(key,key.size(),iv);

    // The StreamTransformationFilter adds padding
    //  as required. ECB and CBC Mode must be padded
    //  to the block size of the cipher.
    StringSource ss1(plain, true,
        new StreamTransformationFilter(e,
            new StringSink(cipher)
        ) // StreamTransformationFilter
    ); // StringSource
}
catch(const CryptoPP::Exception& e)
{
    cerr << e.what() << endl;
    exit(1);
}

StringSource ss2(cipher, true,
    new HexEncoder(
        new StringSink(encoded)
    ) // HexEncoder
); // StringSource
cout << "cipher text [ENCODED] : " << encoded << endl;
encryptedmsg=send_recv(sock, encoded, "Encrypted Message in HEXA format");
}
```

And use send_recv() function to send over the encrypted message to server. And the "encryptedmessage" that sent from "SERVER" will store in a variable **encryptedmsg** (global variable) so in later we can perform decryption on it

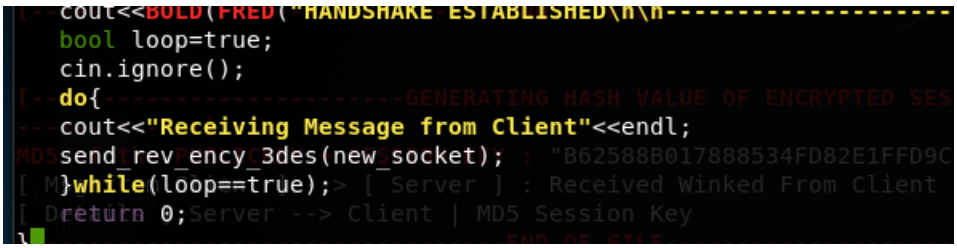
Output of ReverseEncodedSessionKey and HANDSHAKE ESTABLISH, ready to type message and encrypt it send over network.

14. Server then Decrypt the message Mc from client using $K_s1 + K_s2 = K_s$

Back to section 8, after finish sending the "Encrypted SessionKey" and it "hash" to the client.
It will display the "3DES KEY" and "HANDSHAKE ESTABLISHED", and receiving message from client.



On Server's main() it will perform a 'infinite' loop to keep continuous receive ciphertext and decrypt , and input message and encrypt send over network | same goes to client side but will discuss that in later.



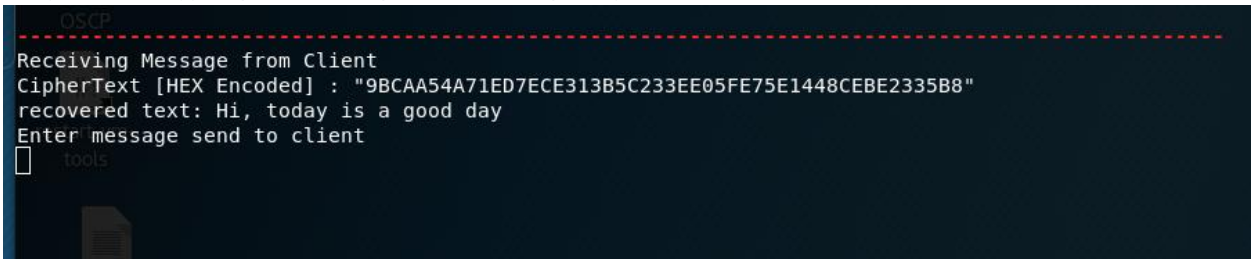
```
void send_rev_ency_3des(int new_socket)
{

   tripledes_decrypt(new_socket,tripledeskey);
   tripleDES_encrypt(tripledeskey,new_socket);
}
```

Let take a quick look at the decryption process .

```
void tripledes_decrypt(int socket,string keys)
{
    int valread;
    char buffer[1024] = {0};
        string compare;
        valread = read( socket , buffer, 1024);
        cout<<"CipherText [HEX Encoded] : \""<<buffer<<"\""<<endl;
        AutoSeededRandomPool prng;
        string rawcipher,decodedkey;
        StringSource ss2(buffer, true,
        new HexDecoder(
                new StringSink(rawcipher)
        ) // HexEncoder
        ); // StringSource
//      bzero(buffer,1024);
        StringSource s(keys, true,(new HexDecoder(
        new StringSink(decodedkey))));
        SecByteBlock key((const byte*)decodedkey.data(), decodedkey.size());
        const byte iv[] = {0x12,0x34,0x56,0x78,0x90,0xab,0xcd,0xef};

try
{
    CBC_Mode< DES_EDE2 >::Decryption d;
    d.SetKeyWithIV(key, key.size(), iv);
    string recovered;
    // The StreamTransformationFilter removes
    //  padding as required.
        string decodedmessage;
        string decodeencryptedmessage;

    StringSource ss3(rawcipher, true,
        new StreamTransformationFilter(d,
            new StringSink(recovered)
        ) // StreamTransformationFilter
    ); // StringSource
    if(recovered=="quit")
     {
        cout<<BOLD(FRED("-----------------------------------\"QUIT\"-------------------------------"))<<endl;
        sendpacket(socket,buffer);
        exit(1);
     }
    cout << "recovered text: " << recovered << endl;
}
catch(const CryptoPP::Exception& e)
{
    cerr << e.what() << endl;
    exit(1);
}
}
```

It will grab to receive the message(ciphertext) which encrypt by the client using 3DES algorithm. After recevied the ciphertext it will store into variable “buffer” and perform hexdecoder and store the “raw ciphertext” into variable rawcipher and try to do decryption using the triple3DES(firstkey + secondkey) , if the recovered message is “quit” then the program will quit, if not then will display the “recoveredtext”

Output of decrypting the message and showing the plaintext



15. Server program: Encrypt the message Ms using 3DES encryption with the agreed session K_s (3DES)
16. Message Ms enters 3DES encryption using CBC Mode

On server side's message encryption, I also uses the same key(firstkey+secondkey 2EDE) decode it using HexDecoder and store the value into decodedkey and convert into key in SecByteBlock data type and the iv in bytes format is const and "fixed".

```
string tripleDES_encrypt(string tripledeskey,int sock)
{
        AutoSeededRandomPool prng;
        string decodedkey;
        StringSource s(tripledeskey, true,(new HexDecoder(
                new StringSink(decodedkey))
        ) // StreamTransformationFilter
    ); // StringSource

SecByteBlock key((const byte*)decodedkey.data(), decodedkey.size());
const byte iv[] = {0x12,0x34,0x56,0x78,0x90,0xab,0xcd,0xef};
string plain;
string cipher, encoded, recovered;

/*********************************\
\*********************************/

do{
cout<<"Enter message send to server"<<endl;
std::getline(std::cin, plain);
if(plain.size()>500)
{
cout<<BOLD(FRED("Message is exceed the length"))<<endl;
}
}while(plain.size()>500);
try
{
    CBC_Mode< DES_EDE2 >::Encryption e;
    e.SetKeyWithIV(key,key.size(),iv);
    // The StreamTransformationFilter adds padding
    //  as required. ECB and CBC Mode must be padded
    //  to the block size of the cipher.
    StringSource ss1(plain, true,
        new StreamTransformationFilter(e,
            new StringSink(cipher)
    ) // StreamTransformationFilter
    ); // StringSource
}
catch(const CryptoPP::Exception& e)
{
    cerr << e.what() << endl;
    exit(1);
}

StringSource ss2(cipher, true,
    new HexEncoder(
        new StringSink(encoded)
    ) // HexEncoder
); // StringSource
cout << "cipher text [ENCODED] : " << encoded << endl;
sendpacket(sock,encoded);
return plain;
}
```

Next, server will enter message and the message will (encrypt it and convert it as hex format) and send over the network to client.

If the input message is more than 500 will display message is exceed the length and ask for reinput the message .

Next it will perform encryption therefore we create an object

```
CBC_MODE<DES_EDE2 >Encryption and setkeywithiv
Next using   StringSource ss1(plain, true,
        new StreamTransformationFilter(e,
            new StringSink(cipher)
```

and the "cipher" variable now do have a rawcipher and we perform hexencoder and store the value into encoded

```
StringSource ss2(cipher, true,
    new HexEncoder(
        new StringSink(encoded)
    ) // HexEncoder
```

```
    ); // StringSource
```

Next, after the message is encrypted and convert to Hex using HexEncoder it will send the ciphertext over the network using sendpacket() function.

```
    sendpacket(sock,encoded);
```

Full code of sendpacket()

```
    void sendpacket(int new_socket,string message)
    {
    send(new_socket , message.c_str() , strlen(message.c_str())+1 , 0 );
    }
```

This function will send the “encrypted message” over the network and at the client side will able to receive the message.

Output Enter Message and encrypted it and send over network

```
Receiving Message from Client
CipherText [HEX Encoded] : "9BCAA54A71ED7ECE313B5C233EE05FE75E1448CEBE2335B8"
recovered text: Hi, today is a good day
Enter message send to client
I'm OSCP. Who are you ?
cipher text [ENCODED] : E5ECE70B7CB12BC27F6F991D35E89E51EED298106F373B44
Receiving Message from Client
```

17. Upon receiving message Ms from server, client can decrypt it using the session K_s

As I mentioned on section 13 we can see that it’s a infinity loop [Image below]

```
    cin.ignore();
    do
    {
    tripleDES_encrypt(tripledeskey,sock);
    tripledes_decrypttext(sock,tripledeskey,encryptedmsg);
    }while(loop==true);
}
```

So after finished encrypt the message and send over the encrypted to server and server decrypt it and server send another “encrypted message”.

Now at client side, after received the message from server, it perform decryption and client will perform the input message and encrypt it just like at section 13 and keep looping until either side type the keyword “quit”.

so now it will perform decryption process, first it will perform Hexdecode on the ciphertext (encryptedmessage) which we can see that the encryptedmessage came from encryptedmsg in main() and the encryptedmsg is came from 3des encryption process which can see in section 13.

```
void tripledes_decrypttext(int socket,string keys,string encryptedmessage)
{
        AutoSeededRandomPool prng;
        string rawcipher,decodedkey;
        StringSource ss2(encryptedmessage, true,
        new HexDecoder(
                new StringSink(rawcipher)
            ) // HexEncoder
        ); // StringSource

        StringSource s(keys, true,(new HexDecoder(
        new StringSink(decodedkey))));

        SecByteBlock key((const byte*)decodedkey.data(), decodedkey.size());
        const byte iv[] = {0x12,0x34,0x56,0x78,0x90,0xab,0xcd,0xef};

try
{
    CBC_Mode< DES_EDE2 >::Decryption d;
    d.SetKeyWithIV(key, key.size(), iv);
    string recovered;
    // The StreamTransformationFilter removes
    //  padding as required.
        string decodedmessage;
        string decodeencryptedmessage;

    StringSource ss3(rawcipher, true,
        new StreamTransformationFilter(d,
            new StringSink(recovered)
        ) // StreamTransformationFilter
    ); // StringSource
     if(recovered=="quit")
     {
        cout<<BOLD(FRED("-----------------------------------\"QUIT\"------------------------------"))<<endl;
        sendpacket(socket,encryptedmessage);
        exit(1);
     }

    cout << "recovered text: " << recovered << endl;
}
catch(const CryptoPP::Exception& e)
{
    cerr << e.what() << endl;
    exit(1);
}
}
```

So after we have finished the Hexdecoder process on cipher, we decode the key(firstkey+ secondkey) into rawkey and store into variable decodedkey.

Next will convert the decodekey from string to a new variable “key” and it datatype is SecBytesBlock. and we create a const bytes iv which is the same value on Server.

Next, will go into the process of tripledes decryption.

Before we go into the process of decryption we have to create a decryption object

```
CBC_Mode< DES_EDE2 >::Decryption d;
```

And setkeywithiv

```
d.SetKeyWithIV(key, key.size(), iv);
```

Now it will perform decryption

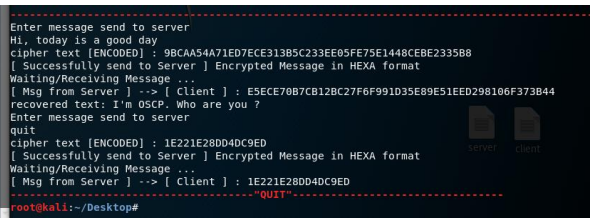
```
    StringSource ss3(rawcipher, true,
            new StreamTransformationFilter(d,
                new StringSink(recovered)
            ) // StreamTransformationFilter
        ); // StringSource
```

It take the rawcipher and using privatekey which is the object name ‘d’ to perform the decryption and store the value into “recovered variable”.

if the recovered text is “quit” then the program will quit, else then will display the “recoveredtext”.

SERVER

CLIENT



Output

Server

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
       inet 10.10.193.99  netmask 255.255.192.0  broadcast 10.10
       inet6 fe80::36db:6c93:714d:f360  prefixlen 64  scopeid 0x
       ether a0:f3:c1:14:f6:b7  txqueuelen 1000  (Ethernet)
       RX packets 34  bytes 7383 (7.2 KiB)
       RX errors 0  dropped 0  overruns 0  frame 0
       TX packets 34  bytes 3845 (3.7 KiB)
       TX errors 0  dropped 0 overruns 0  carrier 0  collisions

root@kali:~/Desktop# ls -l server
-rwxr-xr-x 1 root root 885036 Oct 23 20:28 server
root@kali:~/Desktop#
```

Client

```
       TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
       inet 10.10.225.227  netmask 255.255.192.0  broadcast 10.10.255.255
       inet6 fe80::a03f:ff6f:d5f4:e385  prefixlen 64  scopeid 0x20<link>
       ether 00:11:7f:1a:f8:46  txqueuelen 1000  (Ethernet)
       RX packets 34  bytes 7383 (7.2 KiB)
       RX errors 0  dropped 0  overruns 0  frame 0
       TX packets 33  bytes 3759 (3.6 KiB)
       TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@kali:~/Desktop# ls -l client
-rwxr--r-- 1 root root 859928 Oct 29 15:18 client
root@kali:~/Desktop#
```

```
-rwxr-xr-x 1 root root 885036 Oct 23 20:28 server
root@kali:~/Desktop# ./server
Enter port number to start the listener
5555
[Server] Listening the port 5555 successfully.
```

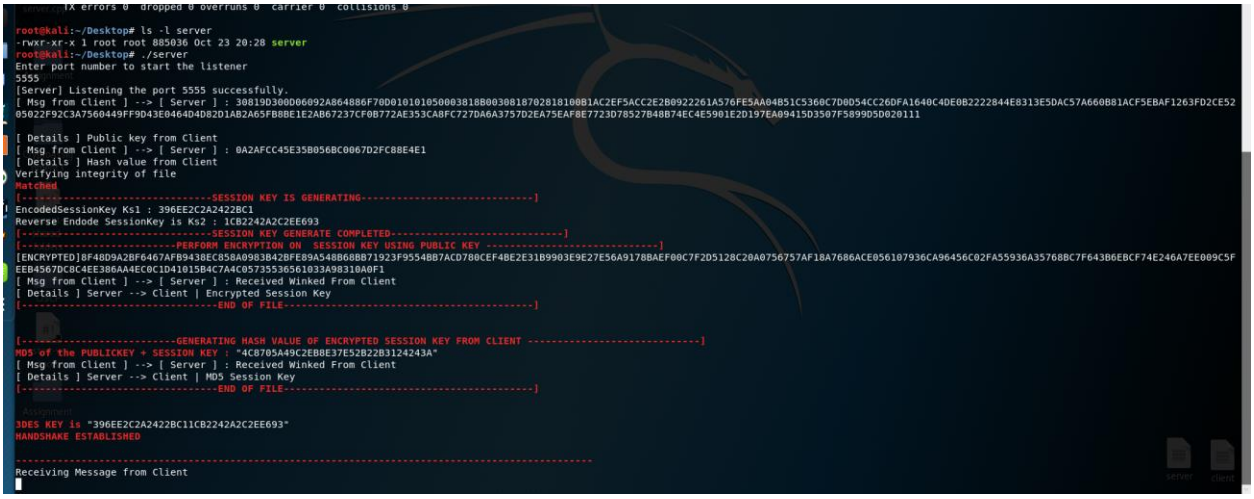
```
-rwxr--r-- 1 root root 859928 Oct 29 15:18 client
root@kali:~/Desktop# ./client
Enter Server IP ADDRESS
10.10.193.99
Enter port number
5555
[System] RSA Key is generating
```

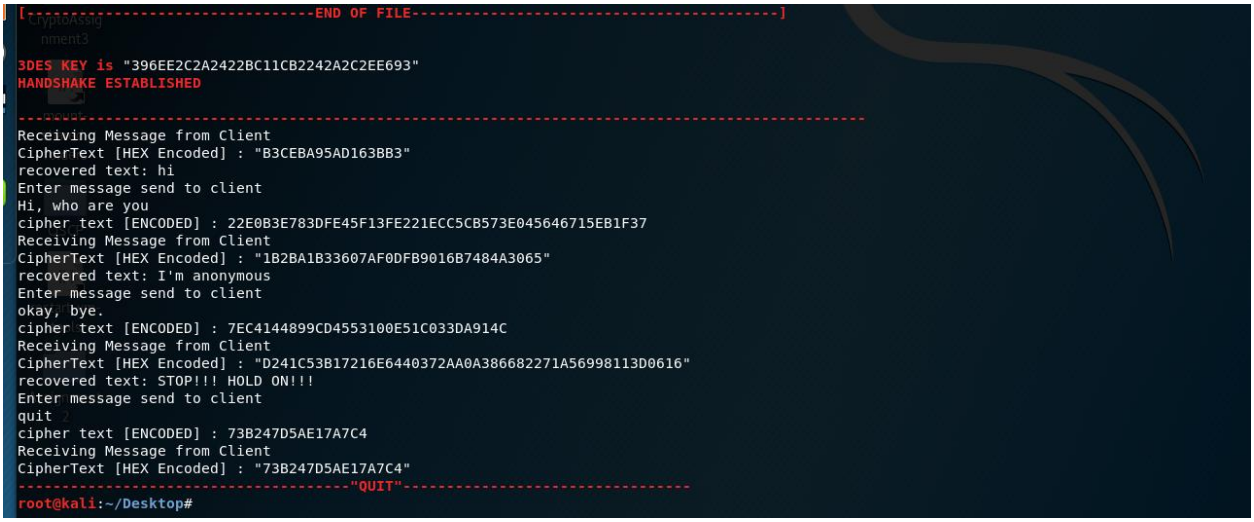
Client

```
[System] RSA Key is generating
[ Successfully send to Server ] Public Key have sent
Waiting/Receiving Message ...
[ Msg from Server ] --> [ Client ] : Received Wink > .. <
----------PUBLIC KEY AND HASH THAT SEND OVER NETWORK----------
Value of Public Key
30819D300D06092A864886F70D0101010500038188003081870281810081AC2EF5ACC2E2B0922261A576FE5AA04B51C5360C7D0D54CC26DFA1640C4DE0B2222844E8313E5DAC57A660B81ACF5EBAF1263FD2CE5205022F92C3A756044
9FF9D43E0464D4D82D1AB2A65FB8BE1E2AB67237CF0B772AE353CA8FC727DA6A3757D2EA75EAF8E7723D78527B48B74EC4E5901E2D197EA09415D3507F5899D5D020111

[MD5]PUBLIC KEY : 0A2AFCC45E35B056BC0067D2FC88E4E1
[ Successfully send to Server ] Hash value "MD5" of public key sent
Waiting/Receiving Message ...
[ Msg from Server ] --> [ Client ] : Received Wink > .. <
----------E O F----------
----------RECEIVED ENCRYPTED SESSION USING PUBLIC KEY AND IT HASH VALUE FROM SERVER----------
---
[ Successfully send to Server ] Encrypted Session Key from Server
Waiting/Receiving Message ...
[ Msg from Server ] --> [ Client ] : 8F48D9A2BF6467AFB9438EC858A0983B42BFE89A548B68BB71923F9554BB7ACD780CEF4BE2E31B9903E9E27E56A9178BAEF00C7F2D5128C20A0756757AF18A7686ACE05610793 6CA96456C02FA55936A35768BC7F643B6EBCF74E246A7EE009C5FEEB4567DC8C4EE386AA4EC0C1D41015B4C7A4C05735536561033A98310A0F1
[ Successfully send to Server ] Hash value "MD5" of Session Key
Waiting/Receiving Message ...
[ Msg from Server ] --> [ Client ] : 4C8705A49C2EB8E37E52B22B3124243A
----------EOF----------
----------Decryption is in progress .. .. .. . . .----------
[ *Session Key found* ] 396EE2C2A2422BC1 | MD5 : 4C8705A49C2EB8E37E52B22B3124243A
----------Process of decryption is completed----------
Verifying integrity of file
Matched
Reverse Endode SessionKey is Ks2 : 1CB2242A2C2EE693
3DES KEY IS "396EE2C2A2422BC11CB2242A2C2EE693"
HANDSHAKE ESTABLISHED

----------
Enter message send to server
```

```
3DES KEY IS "396EE2C2A2422BC11CB2242A2C2EE693"
HANDSHAKE ESTABLISHED

----------
Enter message send to server
hi
cipher text [ENCODED] : B3CEBA95AD163BB3
[ Successfully send to Server ] Encrypted Message in HEXA format
Waiting/Receiving Message ...
[ Msg from Server ] --> [ Client ] : 22E0B3E783DFE45F13FE221ECC5CB573E045646715EB1F37
recovered text: Hi, who are you
Enter message send to server
I'm anonymous
cipher text [ENCODED] : 1B2BA1B33607AF0DFB9016B7484A3065
[ Successfully send to Server ] Encrypted Message in HEXA format
Waiting/Receiving Message ...
[ Msg from Server ] --> [ Client ] : 7EC4144899CD4553100E51C033DA914C
recovered text: okay, bye.
Enter message send to server
STOP!!! HOLD ON!!!
cipher text [ENCODED] : D241C53B17216E6440372AA0A386682271A56998113D0616
[ Successfully send to Server ] Encrypted Message in HEXA format
Waiting/Receiving Message ...
[ Msg from Server ] --> [ Client ] : 73B247D5AE17A7C4
----------"QUIT"----------
root@kali:~/Desktop#
```

Server


root@kali:~/Desktop# ls -l server
-rwxr-xr-x 1 root root 885036 Oct 23 20:28 server
root@kali:~/Desktop# ./server
Enter port number to start the listener
5555
[Server] Listening the port 5555 successfully.
[Msg from Client] --> [Server] : 30819D300D06092A864886F70D0101010500038188003081870281810081AC2EF5ACC2E2B0922261A576FE5AA04B51C5360C7D0D54CC26DFA1640C4DE0B2222844E8313E5DAC57A660BB1ACF5EBAF1263FD2CE52
05022F92C3A7560449FF9D43E0464D4D82D1AB2A65FB8BE1E2AB67237CF0B772AE353CABFC727DA6A3757D2EA75EAF8E7723D78527B48B74EC4E5901E2D197EA89415D3507F5899D5D020111
[Details] Public key from Client
[Msg from Client] --> [Server] : 0A2AFCC45E35B056BC0067D2FC88E4E1
[Details] Hash value from Client
Verifying integrity of file
Matched
[--------------------------------SESSION KEY IS GENERATING---------------------------]
EncodedSessionKey Ks1 : 396EE2C2A2422BC1
Reverse Endode SessionKey is Ks2 : 1CB2242A2C2EE693
[--------------------------------SESSION KEY GENERATE COMPLETED----------------------------]
[-------------------------PERFORM ENCRYPTION ON SESSION KEY USING PUBLIC KEY ----------------------------]
[ENCRYPTED]8F4BD9A2BF6467AFB9438EC858A0983B42BFE89A548B68BB71923F9554BB7ACD780CEF4BE2E31B9903E9E27E56A9178BAEF00C7F2D5128C20A0756757AF18A7686ACE056107936CA96456C02FA55936A35768BC7F643B6EBCF74E246A7EE009C5F
EEB4567DCBC4EE386AA4EC0C1D4101584C7A4C05735536561033A9B31BA0F1
[Msg from Client] --> [Server] : Received Winked From Client
[Details] Server --> Client | Encrypted Session Key
[--------------------------------END OF FILE---]
[-------------------------GENERATING HASH VALUE OF ENCRYPTED SESSION KEY FROM CLIENT ----------------------------]
MD5 of the PUBLICKEY + SESSION KEY : "4CB705A49C2EB8E37E52B22B3124243A"
[Msg from Client] --> [Server] : Received Winked From Client
[Details] Server --> Client | MD5 Session Key
[--------------------------------END OF FILE---]
3DES KEY is "396EE2C2A2422BC11CB2242A2C2EE693"
HANDSHAKE ESTABLISHED
Receiving Message from Client


[--------------------------------END OF FILE---]
3DES KEY is "396EE2C2A2422BC11CB2242A2C2EE693"
HANDSHAKE ESTABLISHED
Receiving Message from Client
CipherText [HEX Encoded] : "B3CEBA95AD163BB3"
recovered text: hi
Enter message send to client
Hi, who are you
cipher text [ENCODED] : 22E0B3E783DFE45F13FE221ECC5CB573E045646715EB1F37
Receiving Message from Client
CipherText [HEX Encoded] : "1B2BA1B33607AF0DFB9016B7484A3065"
recovered text: I'm anonymous
Enter message send to client
okay, bye.
cipher text [ENCODED] : 7EC4144899CD4553100E51C033DA914C
Receiving Message from Client
CipherText [HEX Encoded] : "D241C53B17216E6440372AA0A386682271A56998113D0616"
recovered text: STOP!!! HOLD ON!!!
Enter message send to client
quit
cipher text [ENCODED] : 73B247D5AE17A7C4
Receiving Message from Client
CipherText [HEX Encoded] : "73B247D5AE17A7C4"
----------"QUIT"----------
root@kali:~/Desktop#

USER MANUAL (GITHUB)

README.MD

```
    README.MD
```

GIT CLONE FROM GITHUB

To download the program in Ubuntu or Linux environment

```
    # git clone https://www.github.com/AppleBois/CryptoAssignment4
    Cloning into 'CryptoAssignment4'...
warning: redirecting to
https://github.com/AppleBois/CryptoAssignment4.git/
    remote: Enumerating objects: 29, done.
    remote: Counting objects: 100% (29/29), done.
    remote: Compressing objects: 100% (18/18), done.
    remote: Total 29 (delta 10), reused 24 (delta 7), pack-reused 0
    Unpacking objects: 100% (29/29), done.
```

INSTALLATION

Install the crypto++ library or else might not able to build/compile in later.

```
    #sudo apt-get update
```

Then, issue next command to install crypto++

```
    #sudo apt-get install libcrypto++-dev libcrypto++-doc libcrypto++utils
```

COMMAND TO COMPILE/BUILD

How to compile the source code after issued “git clone” command, and will save as seed_cfb using parameter -o

```
    # cd CryptoAssignment4
    # cd Source
    # g++ -g3 -ggdb -O0 -Wall -Wextra -Wno-unused -o server server.cpp -
lcryptopp
    #++ -g3 -ggdb -O0 -Wall -Wextra -Wno-unused -o client client.cpp -
lcryptopp
```

ADD PRIVILEGE TO THE BINARY FILE

Change program privilege to allow execution after issued “git clone” command

```
# chmod u+x server client
# ls -l
total 1744
-rwxr--r-- 1 root root 859952 Oct 29 19:26 client
-rw-r--r-- 1 root root  12913 Oct 23 18:35 client.cpp
-rwxr--r-- 1 root root 885060 Oct 29 19:26 server
-rw-r--r-- 1 root root  12249 Oct 23 20:28 server.cpp
-rw-r--r-- 1 root root   5855 Oct 29 19:24 Source.7z
# /bin/sh
# ./server
Enter port number to start the listener
^C
# ./client
Enter Server IP ADDRESS
^C
#
```