Name: _____ CS 490 Midterm Project

Fall 2012

Student #: _____ Oct. 6, 2012

CS490 Cryptography Programming Project (120 points)

- ***Total points: 100 points for regular questions plus 20 bonus points for extra questions;***
- ***Due date: take home midterm programming project due at 11:59pm, Oct. 20th 2012***
- ***Exam is open book, open notes, open slides, open homework and lab assignments and in class practice questions and their solutions***
- ***Solutions need to be uploaded on blackboard***
- ***Exam covers questions on implementing DES, AES and RC4 algorithms*** **to secure a communication channel between client and server**

- ***Lectures notes covered by Midterm Exam 1***
  - ***lecture3-introduction to network security***
  - ***lecture4-traditional symmetric cipher***
  - ***lecture5-introduction to modern symmetric key ciphers***
  - ***lecture6-DES***
  - ***lecture8-AES-1***
  - ***lecture9-AES-2***
  - ***lecture10-RC4***

- ***Textbook chapters covered by Midterm Exam 1***
  - ***Chapter 2- Cryptographic Tools***
  - ***Chapter 19- Symmetric Encryption and Message Confidentiality***

- ***Cryptography programming lab covered by Midterm project***
  - ***lecture2- Analysis on Unsecured Communications***
  - ***lecture7- Symmetric-key Ciphers***

Securing UDP Communication Channel

Using a Suite of Encryption/Decryption Algorithms

In this programming project, we will develop a simple secure UDP client/server system, in which two computers (one is installed with the secure UDP client program and the other one is installed with the secure UDP server program) can communicate securely with each other. The encryption and decryption algorithms we will use include the DES algorithm and the RC4 algorithm for requirement, and the AES algorithm for extra bonus. The final submissions for this project include:

- the source code of DESSecureUDPServer.java (or .cpp/.cc if you use c++) and DESSecureUDPClient.java (or .cpp/.cc if you use c++) if we use DES algorithm for encryption/decryption, and two screenshots displaying the encrypted message sent from client to server and from server to client captured by Wireshark, one screenshot displaying the plaintext message and decrypted message on the client side.
- the source code of RC4SecureUDPServer.java (or .cpp/.cc if you use c++) and RC4SecureUDPClient.java (or .cpp/.cc if you use c++) if we use RC4 algorithm for encryption/decryption, and two screenshots displaying the encrypted message sent from client to server and from server to client captured by Wireshark, one screenshot displaying the plaintext message and decrypted message on the client side.
- the source code of AESSecureUDPServer.java (or .cpp/.cc if you use c++) and AESSecureUDPClient.java (or .cpp/.cc if you use c++) if we use AES algorithm for encryption/decryption, and two screenshots displaying the encrypted message sent from client to server and from server to client captured by Wireshark, one screenshot displaying the plaintext message and decrypted message on the client side. (optional)

Please follow strictly the tutorials below to finish this programming project.

1. Motivation of the project

As we have seen in the first lab work (Analysis on Unsecured Communications with Virtual Box) of the course, the forensics analysis on the unsecured UDP communication channel showed that message exchanged between UDP client and UDP server is plaintext and anybody who can intercept this message can find easily the information exchanged between two persons. For example, given two computers A and B, the IP address of A is 10.0.2.15 and IP address of B is 158.65.246.133. A is the UDP client with UDP client program installed and B is UDP server with UDP server program installed. Running Wireshark, we can intercept the message sent from A to B, which is illustrated in Figure 1 and the message sent back from B to A, which is illustrated in Figure 2. By analyzing the message in Figure 1, we can decode that the message sent from client to server is abcdefgh, and by analyzing the message in Figure 2, we can decode that the message sent back from server to client is ABCDEFGH. Therefore, it is pretty easy for us to figure out that the main function of the UDP server program is to capitalize the characters received and then send them back to client, i.e. the server can be called a character capitalization server. Considering the same case applied to an organization required high privacy, such as banks, insurance companies, etc., the unsecured UDP communication is not safe at all and thus motivating us to generate a secured UDP communication channel in which sensitive information can be protected and understood by the two authenticated parties (i.e. holding a shared key) only.

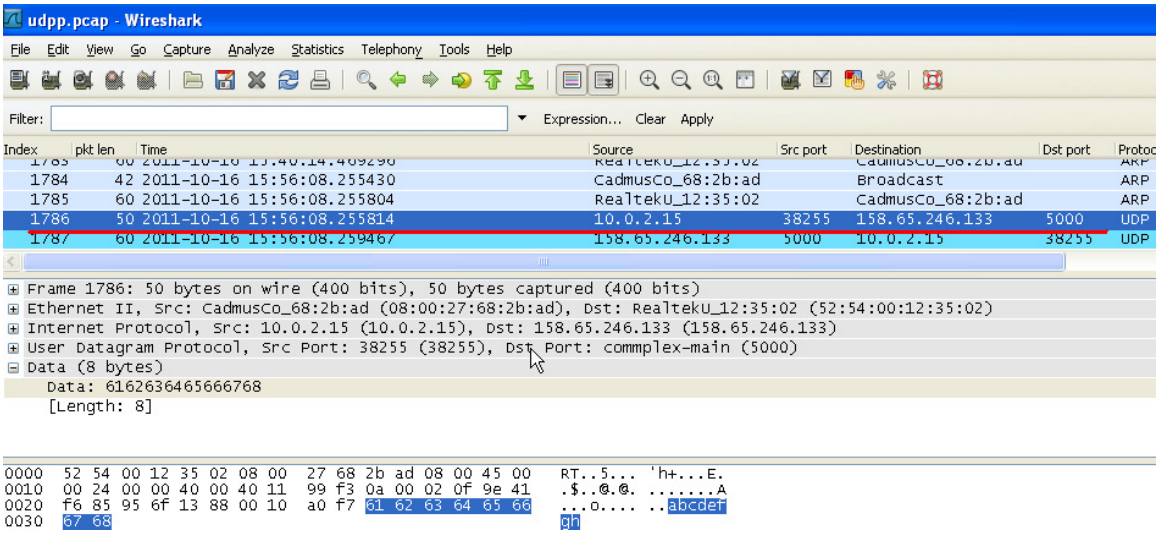


Figure 1. Message sent from UDP client A to UDP server B

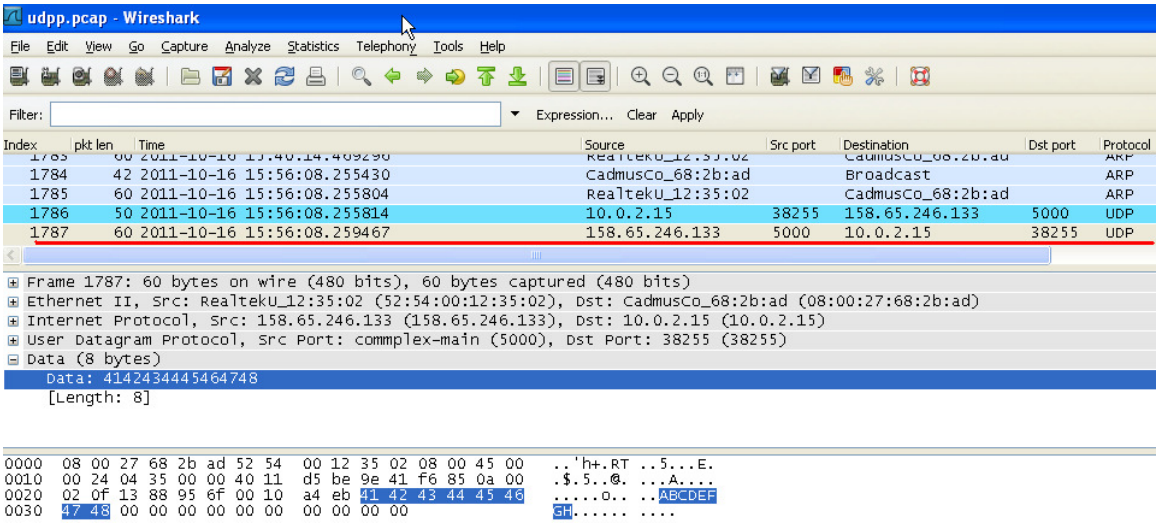


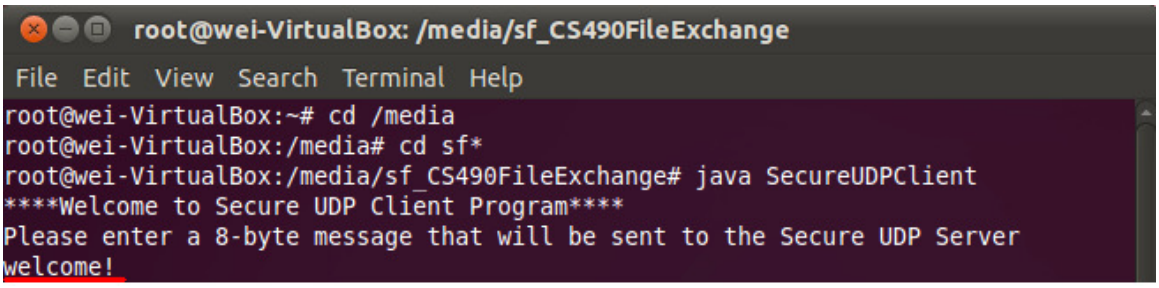
Figure 2. Message sent back from UDP server B to UDP server A

2. Securing the UDP communication

The main task of this project is to protect and secure the UDP communication so two parties using UDP protocol can exchange message privately. In this project we suppose we have the following scenario of UDP communication:

On the client side:

User enters an 8-byte message that will be sent to the Secure UDP Server, for instance, the following screenshot shows that the input message is ***welcome!*** which is 8-byte long including characters 'w','e','l','c','o','m','e','!'



The UDP client program then encrypts the message ***welcome!*** using DES algorithm with the initial key ***iamakey!*** and then sends this message to the server. Since the message is encrypted by DES, the intercepted message by Wireshark is showed as follows:

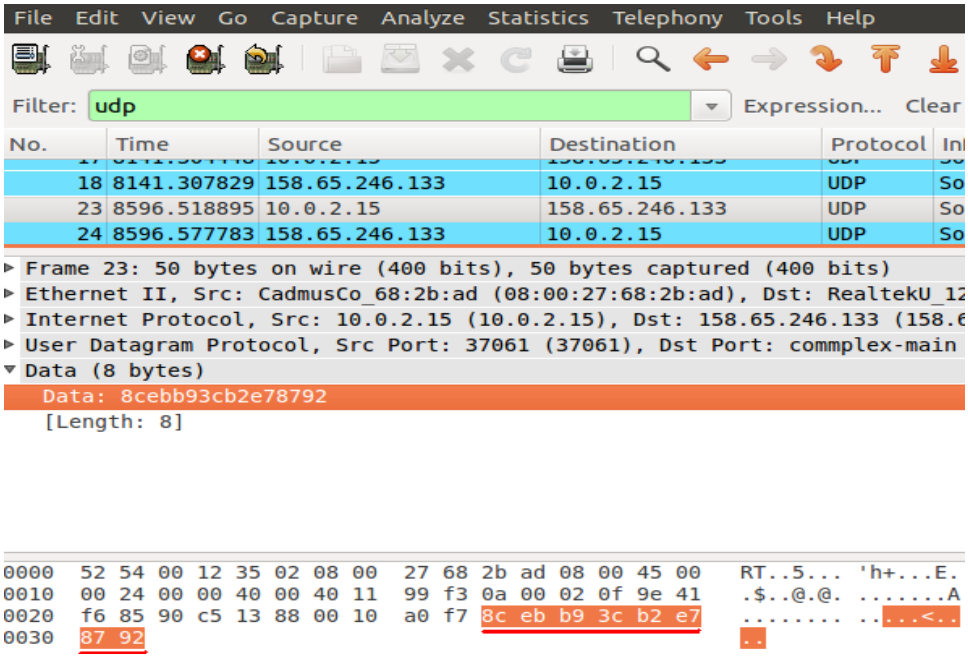


Figure 3. Encrypted message sent from UDP client A to UDP server B

As illustrated in the figure above, it's obvious that the message send from client A to B is 8cebb93cb2e78792 in the format of hexadecimal numbers and intrudy cannot translate it into plaintext at all

On the server side:

Server receives the encrypted message and then decrypts the message into plaintext using the same shared key which is ***iamakey!*** as well. The decrypted message is **welcome!**
Then server capitalizes all the letters and upper case the letters into ***WELCOME!*** and then encrypted the capitalized letters using DES algorithm with the same key ***iamakey!***

After WELCOME! gets encrypted, server then sends it back to client. As illustrated in figure 4 below the intercepted message by Wireshark is c16b61dbe04af8e7

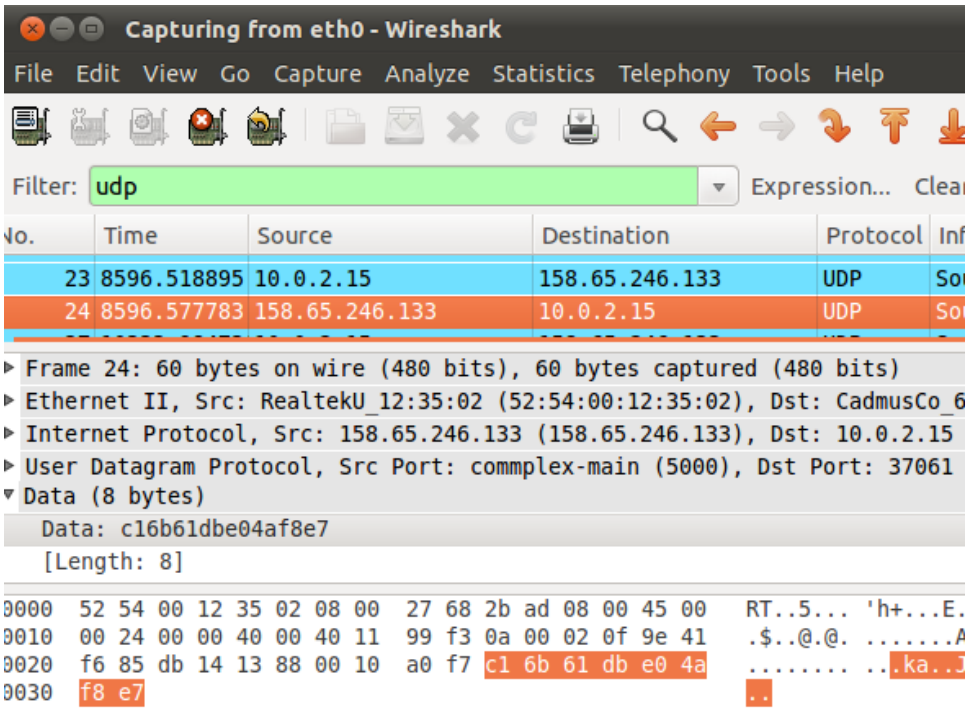


Figure 4. Encrypted message sent from UDP server B to UDP client A

<u>On the client side:</u>

The client, again, receives the encrypted message from server and then decrypts the message using DES algorithm with the key iamakey!
Finally the client displays the decrypted message, which is WELCOME!

3. How to start the project

There are two programs template offered in the project, one is UDP server program called SecureUDPServer.java which can be downloaded at http://www.ece.uvic.ca/~wlu/des/SecureUDPServer.java and the other one is UDP client program called SecureUDPClient.java which can be downloaded at http://www.ece.uvic.ca/~wlu/des/SecureUDPClient.java
The current two programs don't include any encryption and decryption functions and what you can start is to fill in the code of encryption and decryption functions in the programs so that when client and server communicate with each other, their message will be encrypted and transmitted over the Internet. There are two important points about the programs:

(1) The initial key ***iamakey!*** is hard-encoded in the programs . You can change the initial keys to any other 8-byte initial keys, and just make sure both client and server have the same key to encrypt and decrypt the message

(2) You are flexible to change the code of the two programs you downloaded from the link above. Currently the program is hardcoded to set the length of the input message which is 8 bytes, but you are free to change it to 16 bytes, 24 bytes, or any size as long as you remember to use ZERO padding. Since the key is 8-byte long, the input message will be spilt into 8-byte blocks and you can encrypt each block one by one if you have more than 8 bytes message.

4. What to submit and how to grade

Programming Project 1: Implementation of DES algorithm

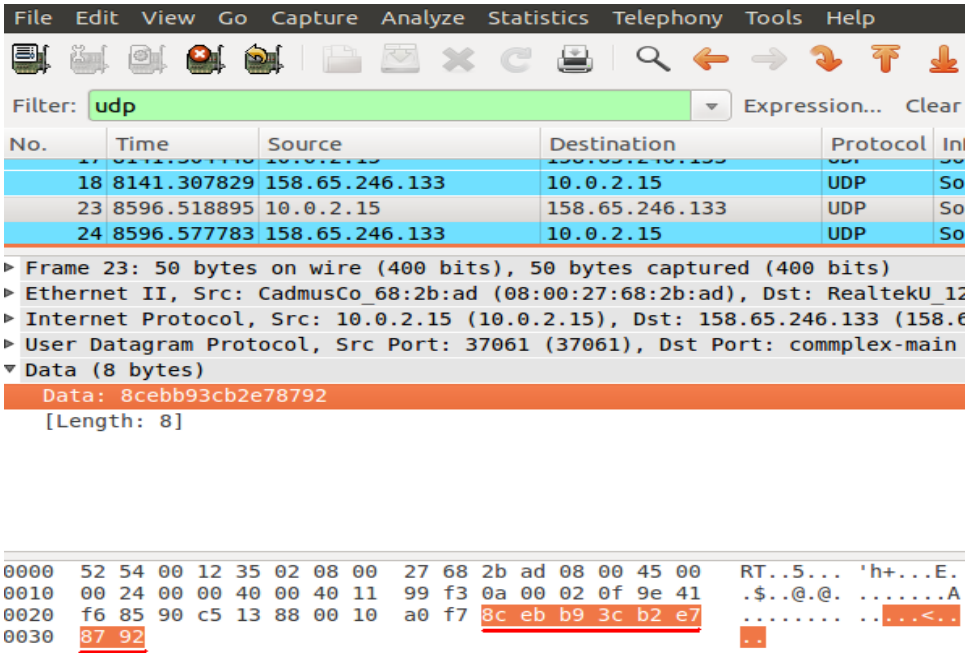
(50 points)

After implementing DES algorithm to encrypt your UDP communication channel, you have to submit 5 files:

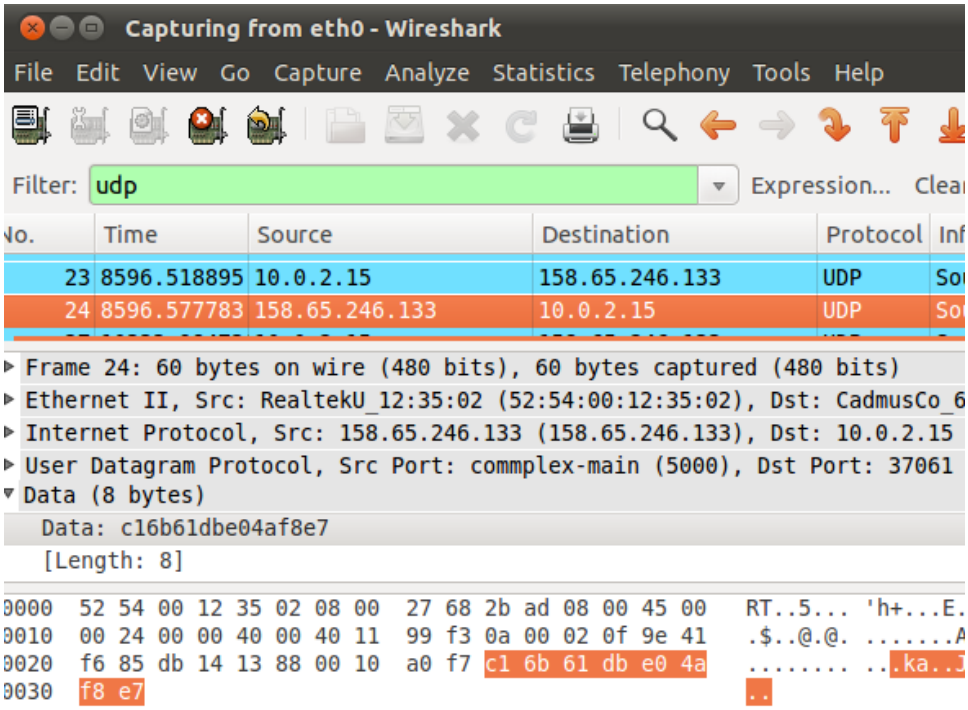
(1) the source code of DESSecureUDPServer.java (or .cc/.cpp if you use c++ programming).

(2) the source code of DESSecureUDPClient.java (or .cc/.cpp if you use c++ programming).

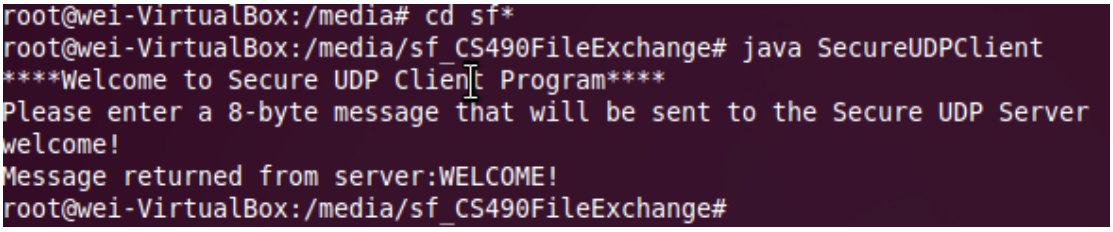
(3) one screenshot displaying the encrypted message sent from client to server captured by Wireshark, for instance, the screenshot below:



(4) one screenshot displaying the encrypted message sent from server to client captured by Wireshark, such as the screenshot below:



(5) one screenshot displaying the plaintext message and decrypted message on the client side.



The grading of the midterm is based on the code comments of your two programs and your screenshots and the grading rubric is lustrated as follows:

(5 points) First comment area contains student name, class and section, correct filename of your assembly program, and a brief description of the program.

(10 points) Implement key generator to generate 16 rounds keys correctly based on the initial key.

(10 points) Screenshot results displaying the encrypted message sent from client to server captured by Wireshark are corrected.

(10 points) Screenshot results displaying the encrypted message sent from server to client captured by Wireshark are corrected.

(10 points) Screenshot results displaying the plaintext message and decrypted message on the client side are corrected.

(5 points) Comments used throughout the program to explain its contents and functionality.

Please note that in this programming question we need to implement the DES encryption and decryption functions on our own, instead of using the existing Java library functions for DES encryption and decryption. However, on the extra programming on AES, you have two choices, (1) you can implement the AES encryption and decryption using the existing JAVA functions or (2) you can implement the AES encryption and decryption functions on your own.

Programming Project 2: Implementation of RC4 algorithm

(50 points)

After implementing RC4 algorithm to encrypt your UDP communication channel, you have to submit 5 files:

(1) the source code of RC4SecureUDPServer.java (or .cc/.cpp if you use c++ programming).

(2) the source code of RC4SecureUDPClient.java (or .cc/.cpp if you use c++ programming).

(3) one screenshot displaying the encrypted message sent from client to server captured by Wireshark.

(4) one screenshot displaying the encrypted message sent from server to client captured by Wireshark.

(5) one screenshot displaying the plaintext message and decrypted message on the client side.

```
root@wei-VirtualBox:/media# cd sf*
root@wei-VirtualBox:/media/sf_CS490FileExchange# java SecureUDPClient
****Welcome to Secure UDP Client Program****
Please enter a 8-byte message that will be sent to the Secure UDP Server
welcome!
Message returned from server:WELCOME!
root@wei-VirtualBox:/media/sf_CS490FileExchange#
```

The grading of the midterm is based on the code comments of your two programs and your screenshots and the grading rubric is lustrated as follows:

(5 points) First comment area contains student name, class and section, correct filename of your assembly program, and a brief description of the program.

(10 points) Implement the key schedule function to create a new s-box for RC4 encryption.

(10 points) Screenshot results displaying the encrypted message sent from client to server captured by Wireshark are corrected.

(10 points) Screenshot results displaying the encrypted message sent from server to client captured by Wireshark are corrected.

(10 points) Screenshot results displaying the plaintext message and decrypted message on the client side are corrected.

(5 points) Comments used throughout the program to explain its contents and functionality.

Please note that in this programming question we need to implement the RC4 encryption and decryption functions on our own, instead of using the existing Java library functions for RC4 encryption and decryption.

Extra Programming Project 3: Implementation of AES algorithm

(20 points)

After implementing AES algorithm to encrypt your UDP communication channel, for considering the extra 20 bonus points, you have to submit 5 files:

(1) the source code of AESSecureUDPServer.java (or .cc/.cpp if you use c++ programming).

(2) the source code of AESSecureUDPClient.java (or .cc/.cpp if you use c++ programming).

(3) one screenshot displaying the encrypted message sent from client to server captured by Wireshark.

(4) one screenshot displaying the encrypted message sent from server to client captured by Wireshark.

(5) one screenshot displaying the plaintext message and decrypted message on the client side.

```
root@wei-VirtualBox:/media# cd sf*
root@wei-VirtualBox:/media/sf_CS490FileExchange# java SecureUDPClient
****Welcome to Secure UDP Client Program****
Please enter a 8-byte message that will be sent to the Secure UDP Server
welcome!
Message returned from server:WELCOME!
root@wei-VirtualBox:/media/sf_CS490FileExchange#
```

The grading of the midterm is based on the code comments of your two programs and your screenshots. There are two potential solutions for this programming project: (1) You can implement the AES encryption and decryption using the existing JAVA functions; Or (2) you can implement the AES encryption and decryption functions on your own.