**Computer Networks**

**(UE17EC351)**

**Communication and Encryption Between Two Different Networks**

Team Members

1. Dhruv Srikanth – PES1201700661
2. Suraj Bidnur – PES1201700145

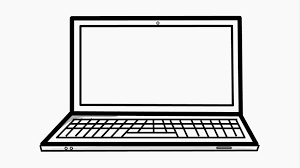
Problem Statement -

1. To achieve communication between two different networks following the client server architecture.
2. Encryption and decryption of the data being communicated.

Block Diagram or Network Diagram –

Encrypted

Encrypted

Decrypted

Decrypted

Internet

Client

Server

Protocol Used -

UDP - User Datagram Protocol

Configuration -

1. Crypto.py -
   * 1. Place file in the same directory as Server.py (at the server end) and Client.py (at the client end).
2. Server.py -
   * 1. First file to be run before any other during execution.
     2. Set ‘port’ variable to any valid port number (Type - Int).
     3. Set ‘host’ variable to the ipv6 address of the network the server will run on (Type - str).
3. Client.py -
   * 1. Set ‘port’ variable to any valid port number (Type - Int). This must be different from the server port number.
     2. Set ‘host’ variable to the ipv6 address of the network the client will run on (Type - str).
     3. Set the first field in the ‘server’ variable server ipv6 address and second field to the server port number. Both found in Server.py

Why ipv6 over ipv4?

Connecting 2 different end systems each connected to different networks could be done using IPv4 also but there are some issues so we used IPv6 instead.

The first problem we came across was public and private IP addresses. Using a private IP address, we could not communicate with a system which was not connected to the same network.

We tried using public IPv4 but then we had to have access to the router settings and set up port forwarding on the router at each end. This is doable but not convenient.

Plus, in this day and age, IPv4 is limited.

So, we used IPv6 instead, which is so large that everyone in the world can get a unique IPv6 address. This overcomes the problem of public and private IP addresses.

Features:

Encryption on the client end and decryption on the server end.

The following cryptographic ciphers are available to the client for encryption -

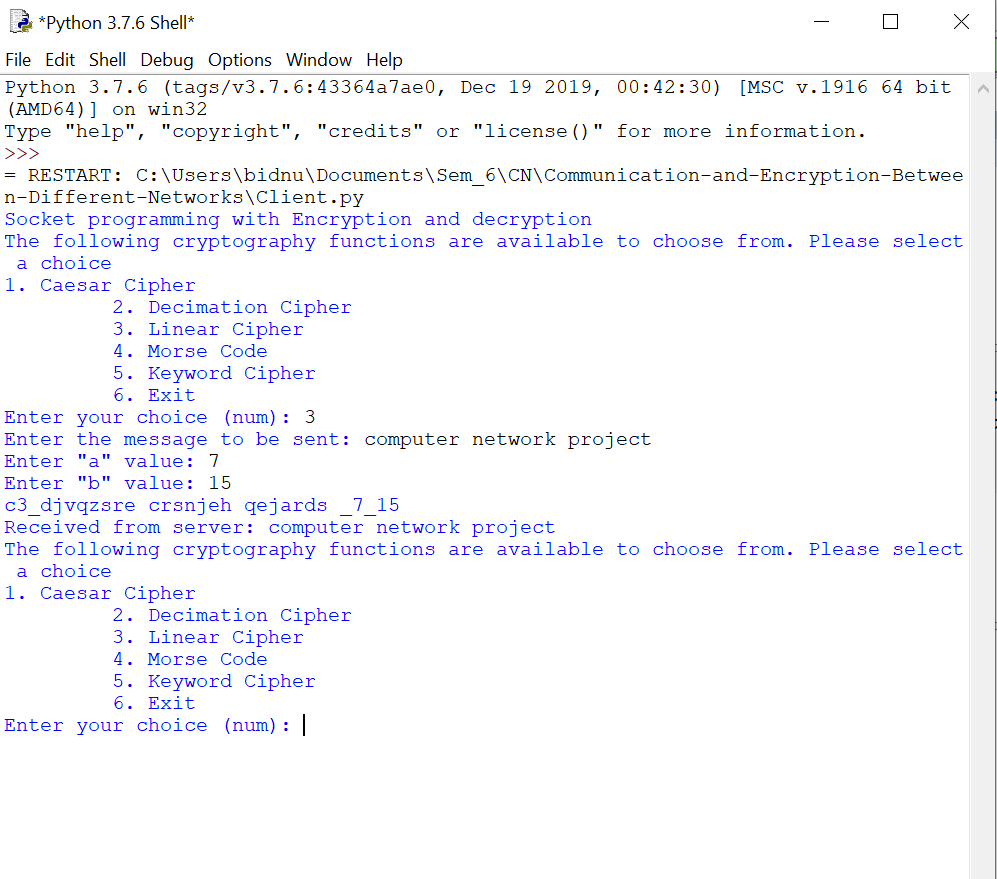
1. Caesar cipher
2. Decimation cipher
3. Linear cipher
4. Morse
5. Keyword

How to run?

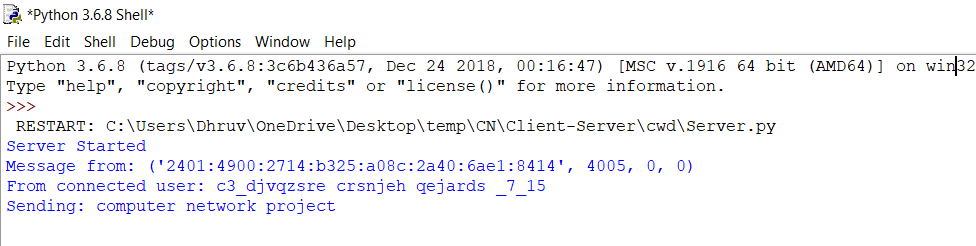
1. Follow and complete steps in Configuration tab.
2. Run Server.py code.
3. Run Client.py code.

Output -

1. Command prompt -
   * 1. Client.py –

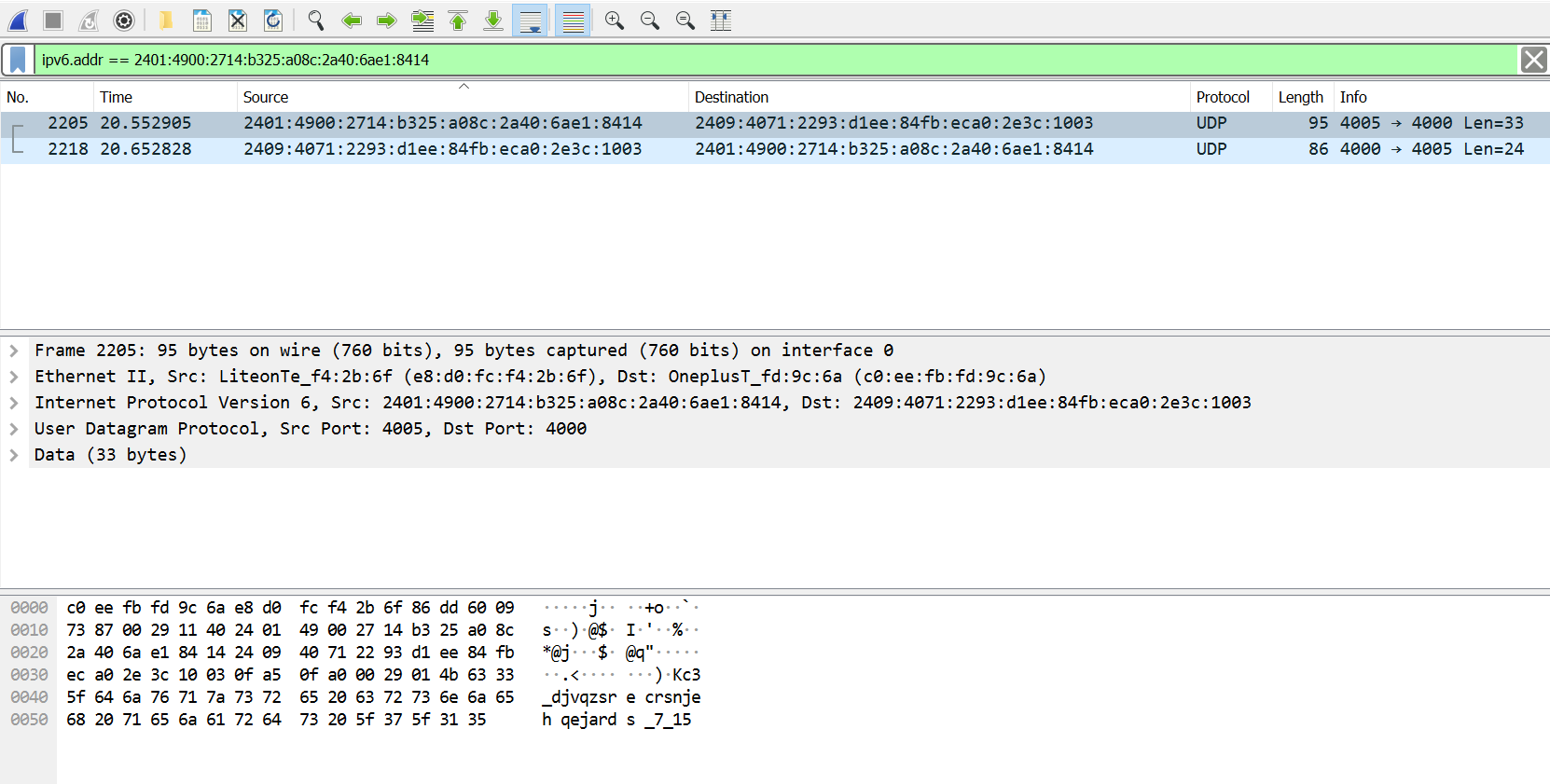


* + 1. Server.py -

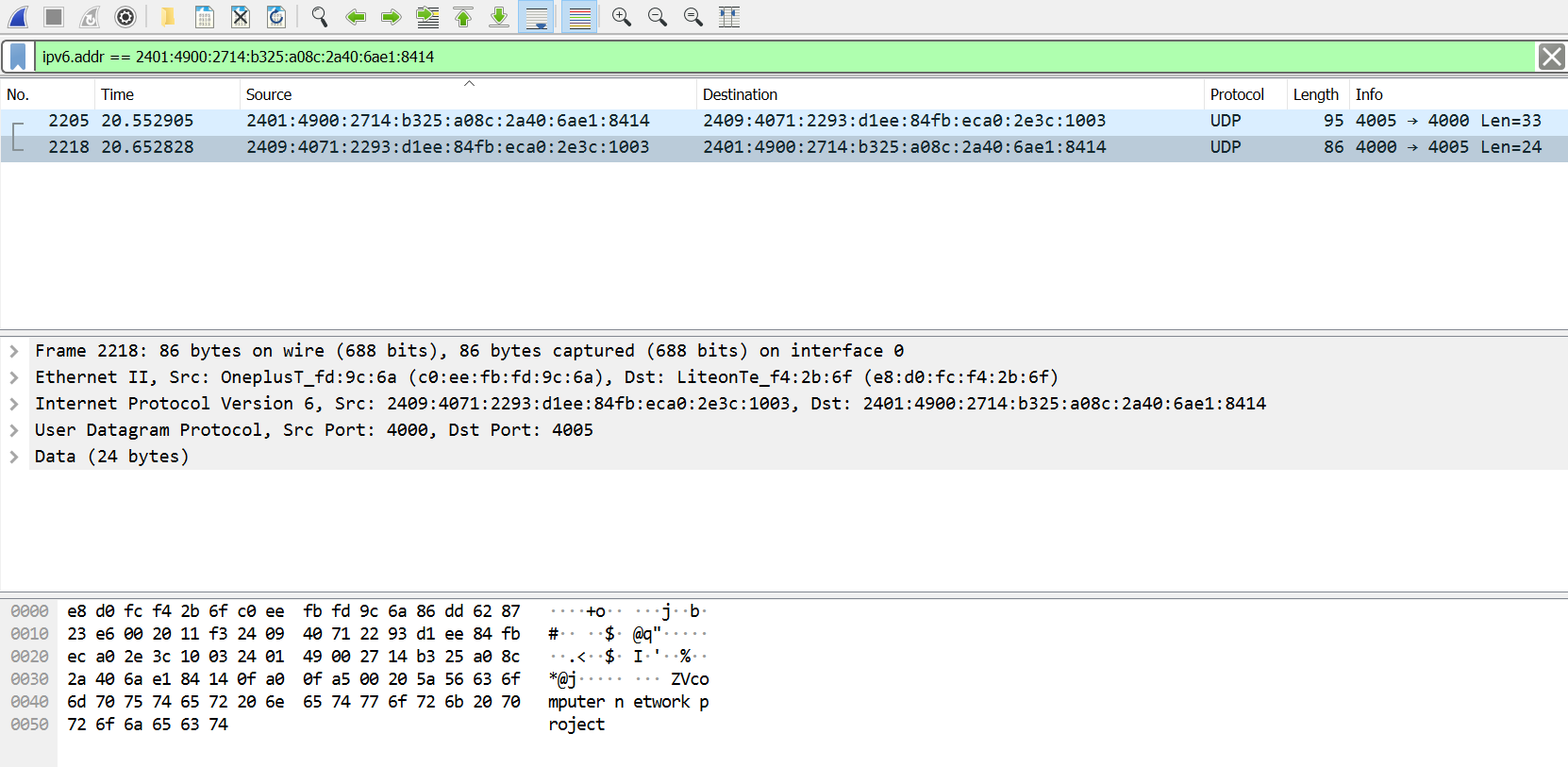


1. Wireshark –
   * 1. Client.py –

Client sends encrypted data

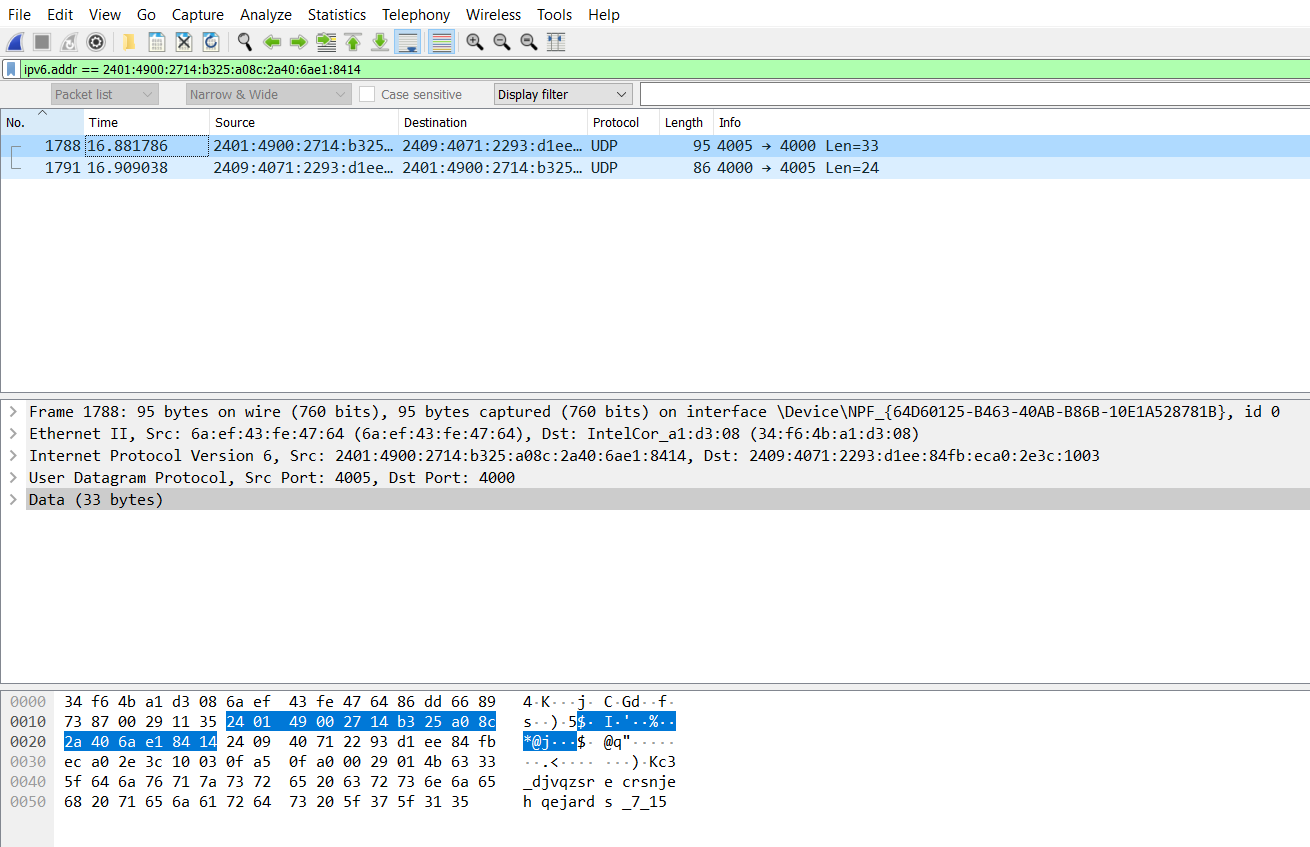


Client receives decrypted data from server

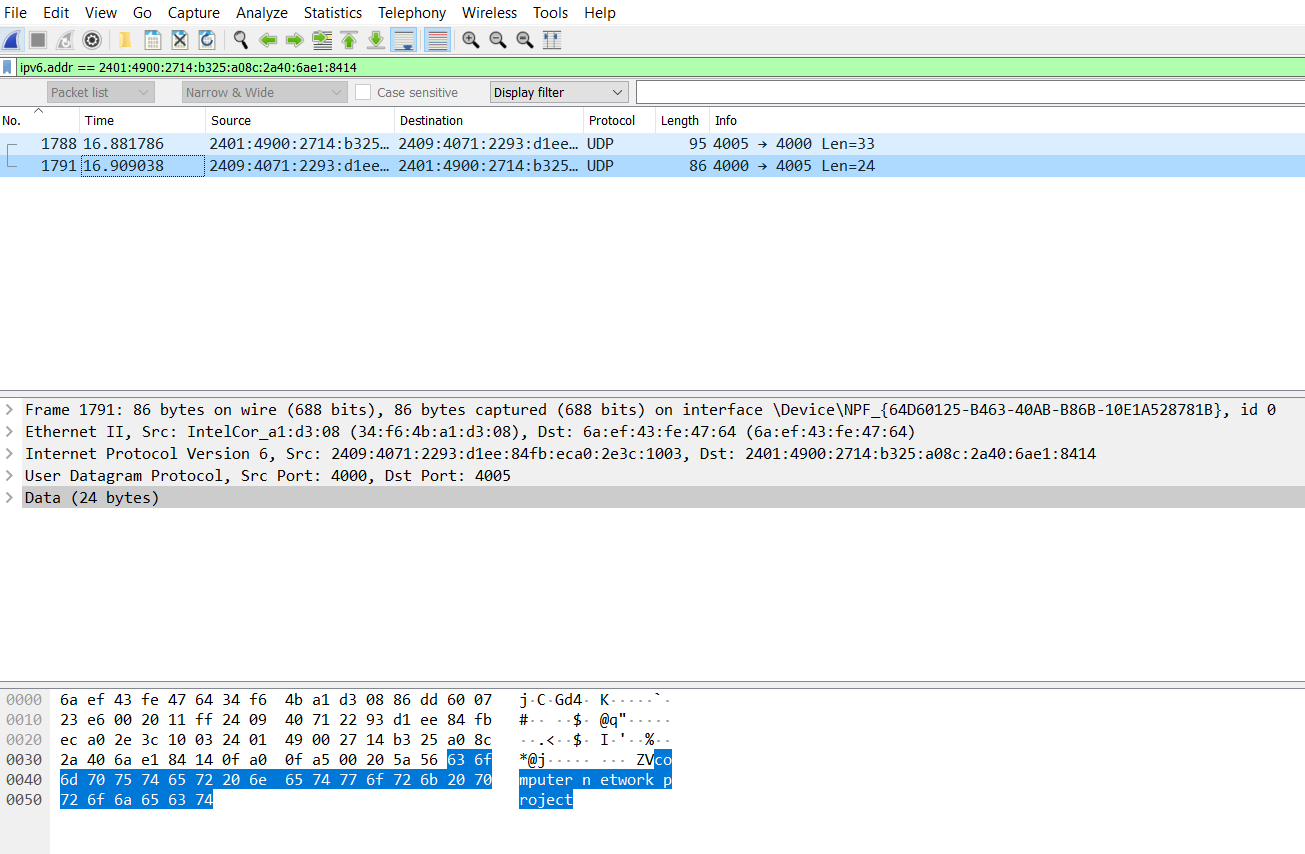


* + 1. Server.py –

Server receives encrypted data -



Server sends back decrypted data to client -



Discussion of results –

The data is encrypted at the client side and then transmitted using UDP (User Datagram Protocol) to the server side. At the server end, the data is decrypted. Once decrypted, a confirmation of the same data is sent back to the client end using UDP and then displayed at the client end. The same can be seen in the output screenshots of both Wireshark and the python runtime window output.

This tells us that since IPv6 is so large that everyone can get a unique IP address, we can communicate with the other device given that we know the correct IPv6 address.