Secure File Transfer over LAN

Secure File Transfer over LAN

GROUP 1

S.NO	NAME	ENROLLMENT NO.	CONTRIBUTION PAGES.
1.	Aayushi Jain	20535001	9 – 11
2.	Ayush Aggarwal	20535008	19 – 22
3.	Km Khushbu	20535014	23 - 29
4.	Mohammad Nazim	20535017	3 – 8
5.	Preeti	20535021	14 – 16
6.	Pushpamanjari	20535022	17– 18
7.	Tanmay Narayan Jawkhede	20535031	11 – 13

INDEX

Team member's contribution:

1. **Aayushi Jain (20535001) :**
    - Sender module
        - Connection establishment
        - sending encrypted file

2. **Ayush Aggarwal (20535008):**
    - Network analysis.
    - Packet capturing using Wireshark.

3. **Km Khushbu (20535014):**
    - File Service server Module.

4. **Mohammad Nazim (20535017):**
    - GUI.
    - Framework design.

5. **Preeti (20535021):**
    - Encrypting and Decrypting file using AES .

6. **Pushpamanjari (20535022):**
    - Encrypting and Decrypting AES key using RSA algorithm.

7. **Tanmay Narayan Jawkhede (20535031):**
    - Receiver module
        - Connection establishment
        - Receiver permission verification
        - Passcode verification
        - Accepting encrypted file and Store on Receiver End in its Original form.

Problem Statement:

An organization needs an application which can help their employees to transfer files between them securely on the same network. Develop an application using socket programming to send files between two machines and secure the data transfer using a strong encryption algorithm. Capture these packets using a sniffing tool like wireshark and show that data transfer is secure.

Importance of Security

Network security is one of the most important aspects to consider when working over the internet, LAN or other method, no matter how small or big your business is. While there is no network that is immune to attacks, a stable and efficient network security system is essential to protecting client data. A good network security system helps business reduce the risk of falling victim of data theft and sabotage. Network security helps protect your workstations from harmful spyware. It also ensures that shared data is kept secure. Network security infrastructure provides several levels of protection to prevent MiM attacks by breaking down information into numerous parts, encrypting these parts and transmitting them through independent paths thus preventing cases like eavesdropping.

Types of Attacks

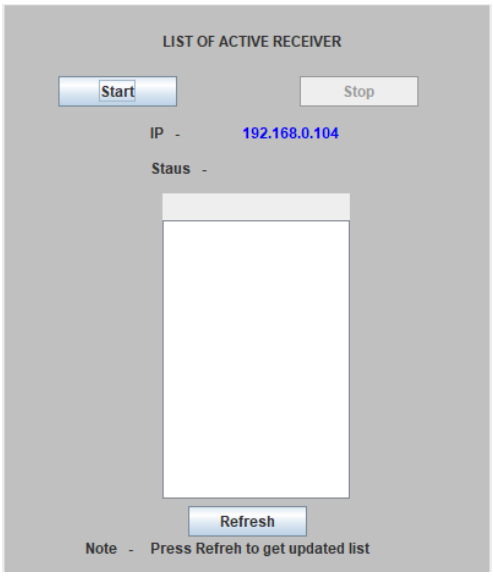
- Man-in-the-middle attack
- Phishing and spear phishing attacks
- Drive-by attack
- Password attack
- SQL injection attack
- Cross-site scripting (XSS) attack
- Eavesdropping attack
- Birthday attack
- Malware attack

Application User Interface:

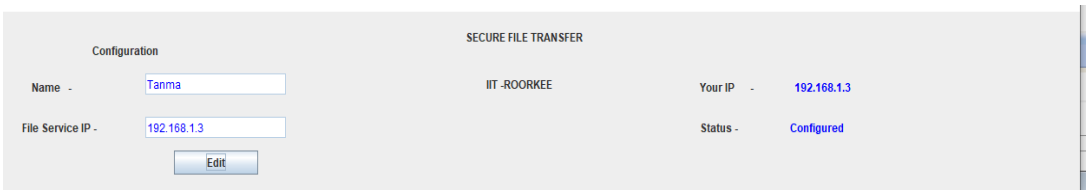
The interactive GUI enables you to send/receive files while monitoring active receivers. The GUI enables a user to easily handle the flow of control like sending a request, accepting/rejecting a request, selecting a registered user, choosing a file etc.

To understand all the functionalities of software, we can divide the software in two frames namely 'file service server' and 'main application'. Their respective functionalities are described below.

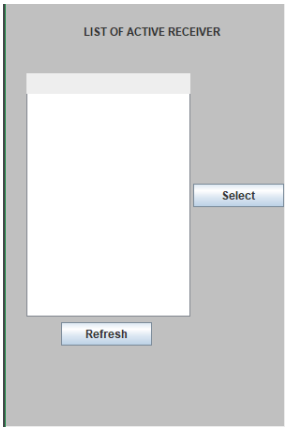
1. **File service server:** This frame runs is built with the purpose of monitoring all the active receivers. This frame is intended to run on a high priority computer like manager's computer. Who might want to monitor active receivers.
   A socket is created on port number **9200 by receiver,** the receiver sends its IP address along with its registered name in the form of an object. This object is received by listening to the same port and adds the receiver to the "list of receivers".
   By default, the server is started on its own when the application is opened.
   It has 2 buttons.
   a. Start: This button starts the server.
   b. Stop: It clears the list and closes the socket.



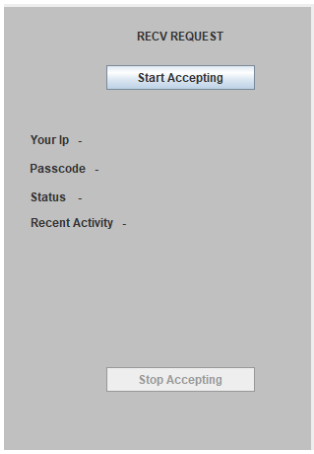
2. Main Application: This frame has following panels:
   a. CONFIGURATION: Here the receiver/sending enters the IP address of file service server along with its name to get itself registered with the file service server.



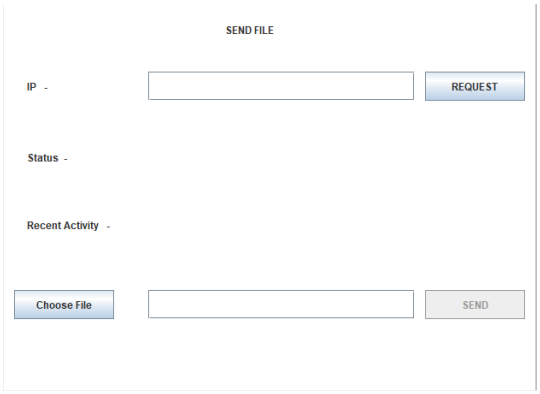
b. LIST OF ACTIVE RECEIVERS: This panel updates the list of receivers fetched from 'file service server' when clicked on 'refresh' button. We can also select one of the registered uses for transferring data. The IP address of that user will be fetched.



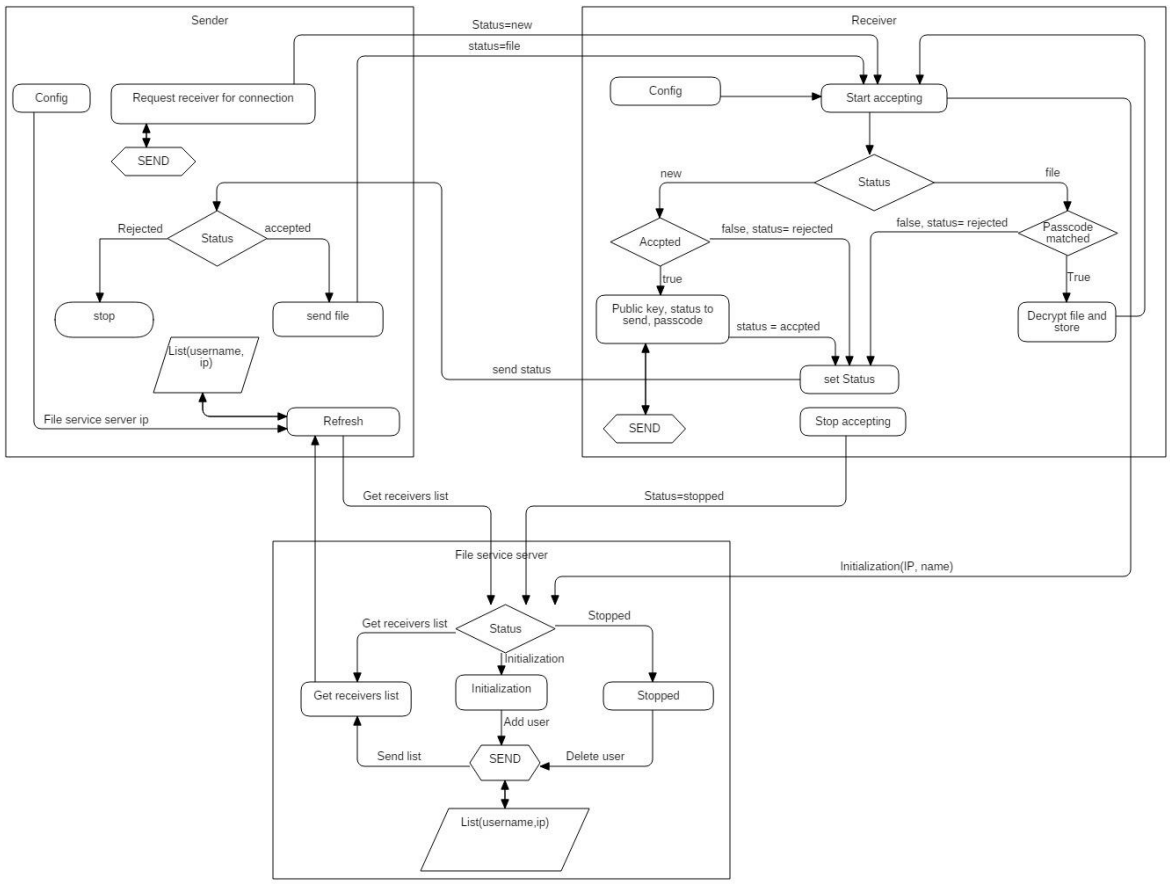
c. RECV REQUEST: This panel have 3 buttons:
   i. Start accepting: This button is used to make the user as "receiver", It creates a socket on port number **9199** for file transfer and sends the configured data to 'file service server' by creating another socket at port number **9200** .
   Based on 'Status' sent by sender, it either sends file (when accepted the request to) or sends passcode+publicKey.
   ii. Accept: when the sender sends request to receiver, this button gets activated. To accept the request receiver should click on accept to establish connection for file transfer.
   iii. Reject: this button will deny the request of sender to establish a connection for file transfer.



d. SEND FILE: This panel have 3 buttons:
    i. REQUEST: This is used by the sender to send a request to receiver (IP address of receiver is fed in the text field besides).
    ii. Choose file: This button pop ups a window to select a file to be sent.
    iii. Send: This button is used to finally encrypt and send the file. This button gets activated only when the receiver accepts the request.

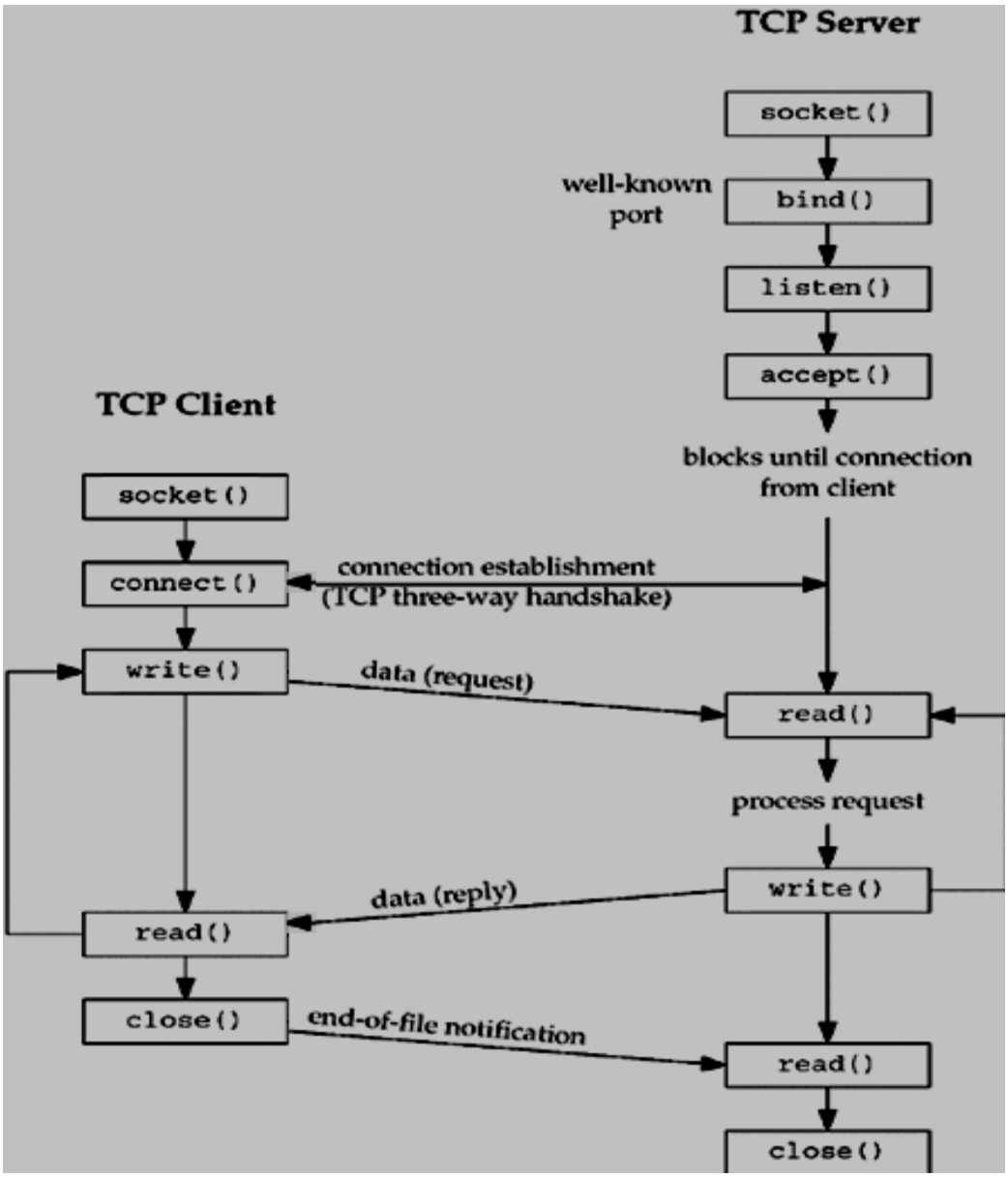


Flow Diagram:



Socket Programming

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.



Socket Programming in Java

Java API networking package (java.net) takes care of most of the stuff needed to establish a connection, making network programming very easy for programmers.

Client Side Programming

Establish a Socket Connection

To connect to other machine we need a socket connection. A socket connection means the two machines have information about each other's network location (IP Address) and TCP port.The java.net.Socket class represents a Socket. To open a socket:

```
Socket socket = new Socket("127.0.0.1", 5000)
```

- First argument – IP address of Server. (127.0.0.1 is the IP address of localhost, where code will run on single stand-alone machine).
- Second argument – TCP Port. (Just a number representing which application to run on a server. For example, HTTP runs on port 80. Port number can be from 0 to 65535)

Communication

To communicate over a socket connection, streams are used to both input and output the data.

Closing the connection

The socket connection is closed explicitly once the message to server is sent. In the program, Client keeps reading input from user and sends to the server until "Over" is typed.

Server-Side Programming

Create Socket Server

Creating Socket Server is Step at which we bind socket to specified port .

Establish a Socket Connection

To write a server application two sockets are needed.

- A ServerSocket which waits for the client requests (when a client makes a new Socket())
- A plain old Socket socket to use for communication with the client.

Communication

getOutputStream() method is used to send the output through the socket.

Close the Connection

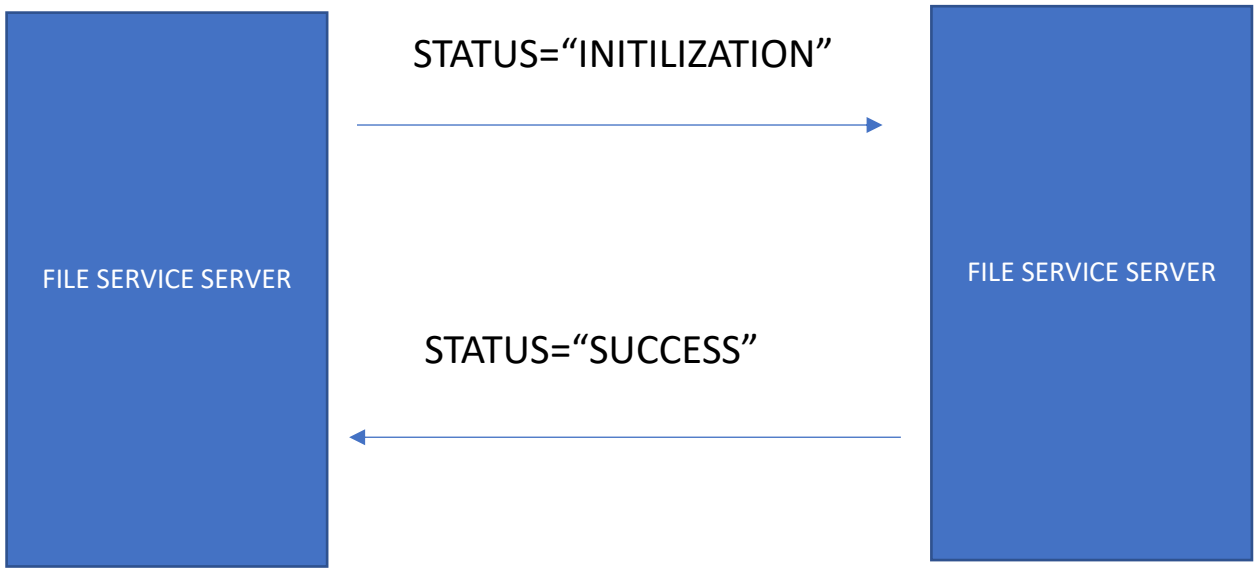
After finishing, it is important to close the connection by closing the socket as well as input/output streams.

- Receiver application makes a ServerSocket on a specific port which is 9199. This starts our Receiver listening for sender requests coming in for port 5000
- Then Server makes a new Socket to communicate with the client.

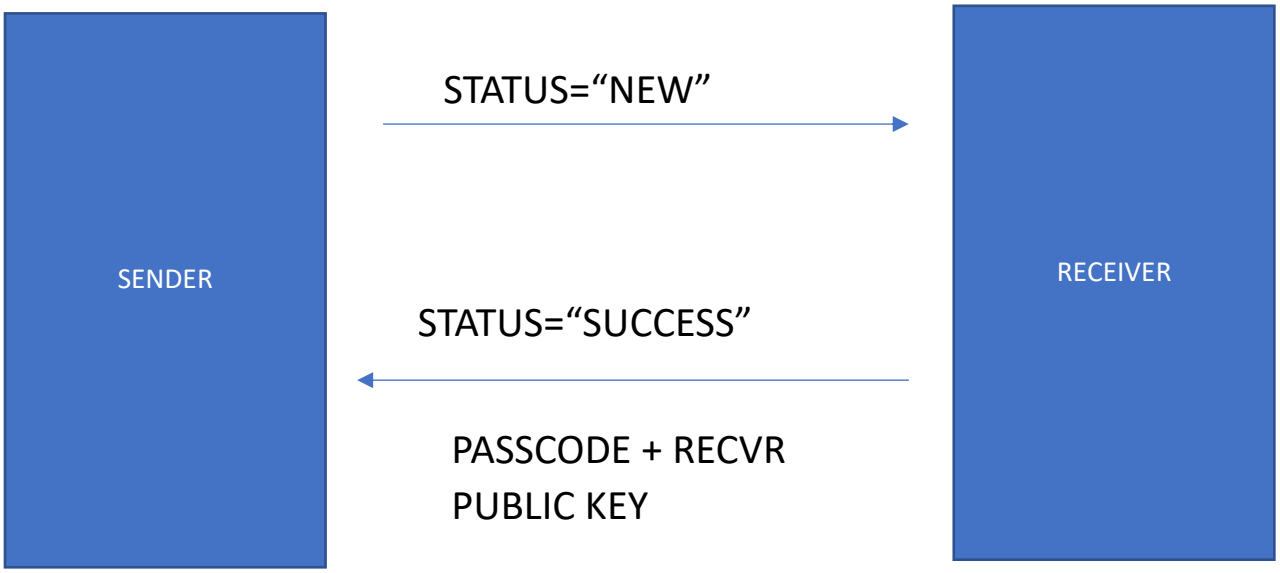
```
socket = server.accept()
```

- The accept() method blocks(just sits there) until a client connects to the server.
- Then we take input from the socket using getInputStream() method. Our Server keeps receiving messages until the Client sends "Over".
- After we're done we close the connection by closing the socket and the input stream.

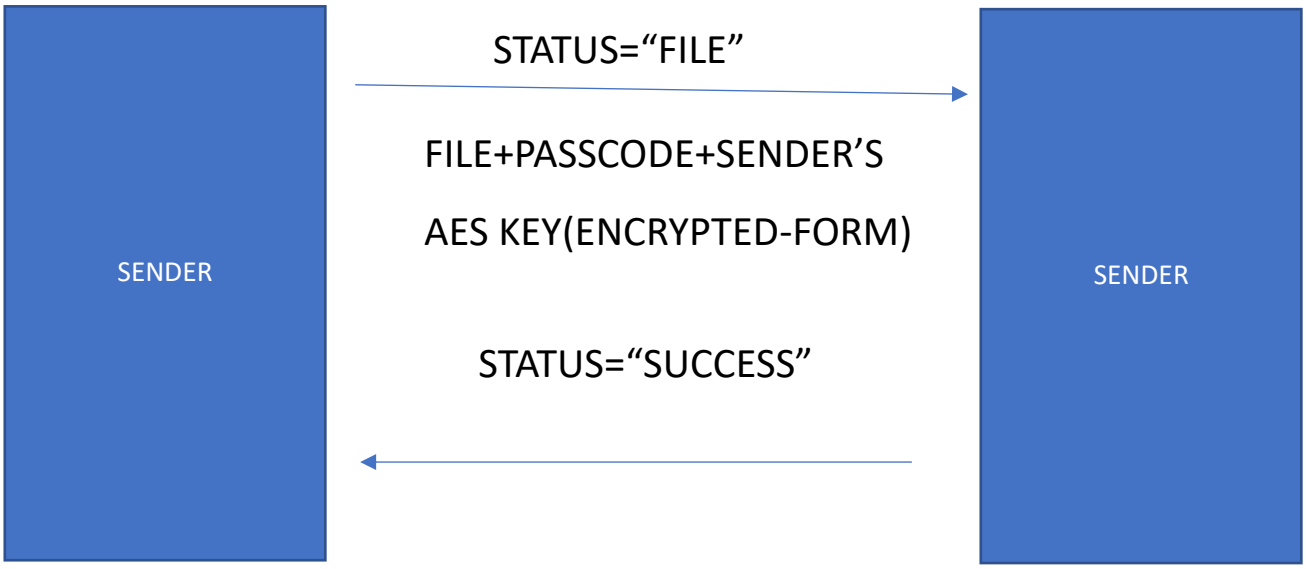
"Receiver Registration"



"Sender Request"



“File Accepting ”



Cryptography

A cryptosystem is an implementation of cryptographic techniques and their accompanying infrastructure to provide information security services. A cryptosystem is also referred to as a cipher system.

The various components of a basic cryptosystem are Plaintext, Encryption Algorithm, Ciphertext, Decryption Algorithm, Encryption Key and, Decryption Key.

Where,

- **Encryption Key** is a value that is known to the sender. The sender inputs the encryption key into the encryption algorithm along with the plaintext in order to compute the cipher text.
- **Decryption Key** is a value that is known to the receiver. The decryption key is related to the encryption key, but is not always identical to it. The receiver inputs the decryption key into the decryption algorithm along with the cipher text in order to compute the plaintext.

Fundamentally there are two types of keys/cryptosystems based on the type of encryption-decryption algorithms.

Symmetric Key Encryption

The encryption process where same keys are used for encrypting and decrypting the information is known as Symmetric Key Encryption.

The study of symmetric cryptosystems is referred to as symmetric cryptography. Symmetric cryptosystems are also sometimes referred to as secret key cryptosystems.

Following are a few common examples of symmetric key encryption −

- Digital Encryption Standard (DES)
- Triple-DES (3DES)
- IDEA
- BLOWFISH

Asymmetric Key Encryption

The encryption process where different keys are used for encrypting and decrypting the information is known as Asymmetric Key Encryption. Though the keys are different, they are mathematically related and hence, retrieving the plaintext by decrypting cipher text is feasible.

Encryption Algorithms used in the project:

AES :

We have used this algorithm to Encrypt the file the we want to transfer.

AES (Advanced Encryption Standard) is a strong symmetric encryption algorithm. A secret key is used for the both encryption and decryption of data. Only someone who has access to the same secret key can decrypt data. AES encryption provides strong protection to data.

Java provides a number of helper classes for AES encryption such as Cipher (for encryption/decryption), SecretKey (represents the shared secret key) and KeyGenerator (generates the shared secret key). Also note that both secret key and encrypted data is binary data and hence cannot be accessed directly.

The features of AES are as follows −

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

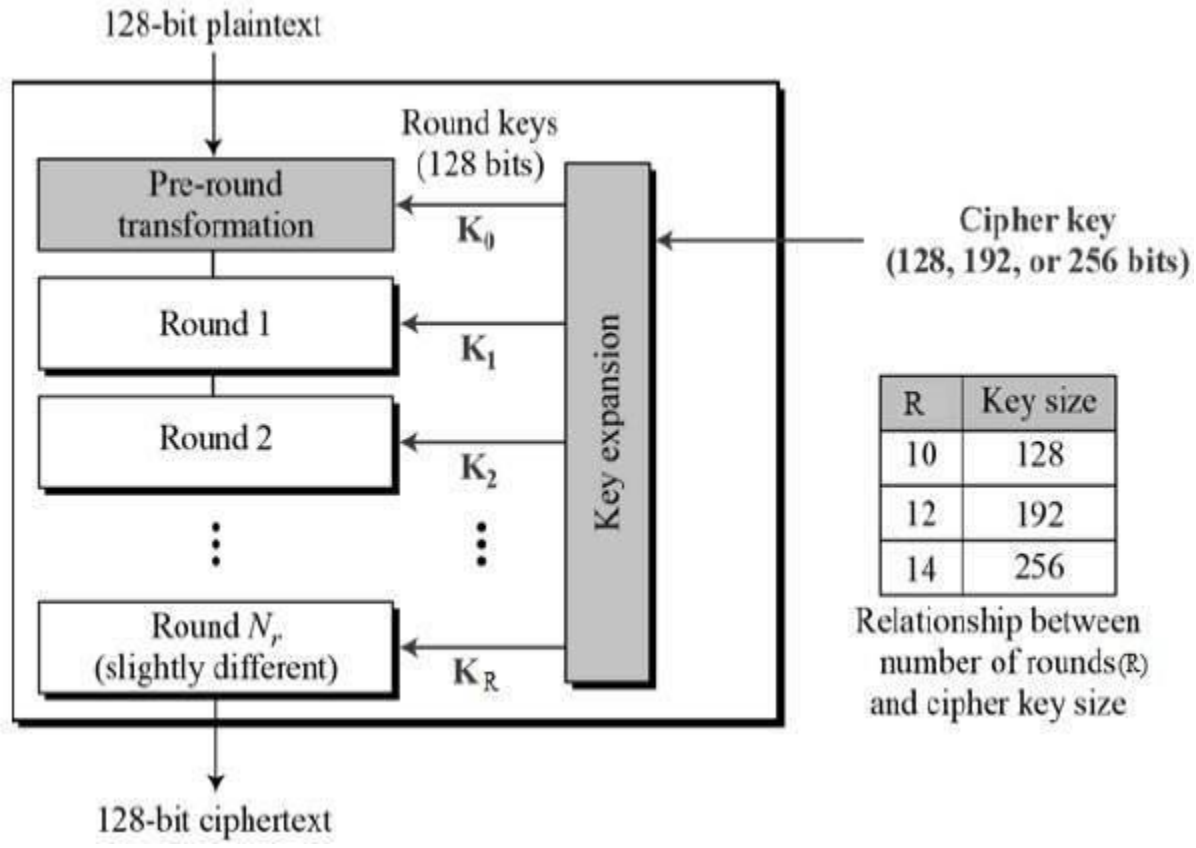
Operation of AES

AES is an iterative rather than Feistel cipher. It is based on 'substitution–permutation network'. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).

Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix −

Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

The schematic of AES structure is given in the following illustration −

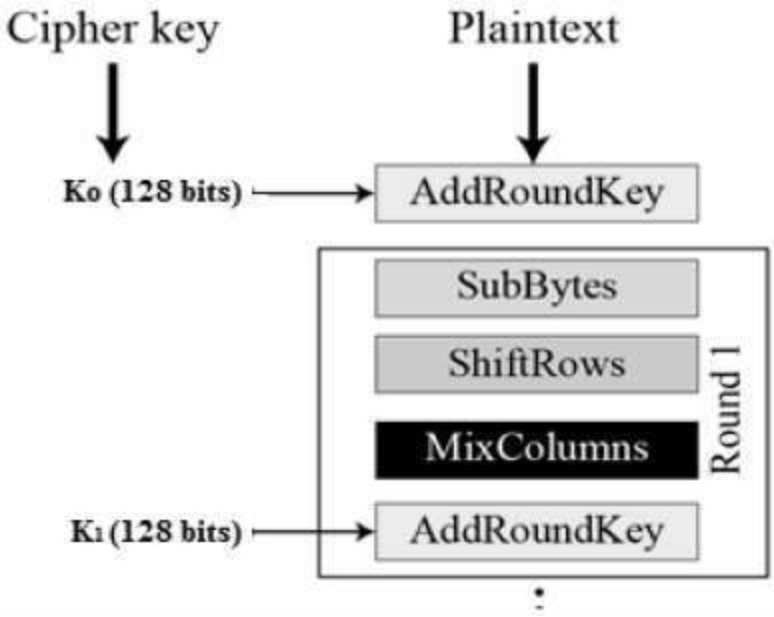


R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

Encryption Process

Here, we restrict to description of a typical round of AES encryption. Each round comprise of four sub-processes. The first round process is depicted below −



Algorithm

1. KeyExpansion – round keys are derived from the cipher key using the AES key schedule. AES requires a separate 128-bit round key block for each round plus one more.
2. Initial round key addition:
    1. AddRoundKey – each byte of the state is combined with a byte of the round key using bitwise xor.
3. 9, 11 or 13 rounds:
    1. SubBytes – a non-linear substitution step where each byte is replaced with another according to a lookup table.
    2. ShiftRows – a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
    3. MixColumns – a linear mixing operation which operates on the columns of the state, combining the four bytes in each column.
    4. AddRoundKey
4. Final round (making 10, 12 or 14 rounds in total):
    1. SubBytes
    2. ShiftRows
    3. AddRoundKey

AES Analysis

In present day cryptography, AES is widely adopted and supported in both hardware and software. Till date, no practical cryptanalytic attacks against AES has been discovered.

Additionally, AES has built-in flexibility of key length, which allows a degree of 'future-proofing' against progress in the ability to perform exhaustive key searches.

However, just as for DES, the AES security is assured only if it is correctly implemented and good key management is employed.

RSA

We have used this algorithm to encrypt the AES key

RSA algorithm is a public key encryption technique and is considered as the most secure way of encryption. It was invented by Rivest, Shamir and Adleman in year 1978 and hence name **RSA** algorithm.

Algorithm

The RSA algorithm holds the following features −

- RSA algorithm is a popular exponentiation in a finite field over integers including prime numbers.
- The integers used by this method are sufficiently large making it difficult to solve.
- There are two sets of keys in this algorithm: private key and public key.

You will have to go through the following steps to work on RSA algorithm −

Step 1: Generate the RSA modulus

The initial procedure begins with selection of two prime numbers namely p and q, and then calculating their product N, as shown −

N=p*q

Here, let N be the specified large number.

Step 2: Derived Number (e)

Consider number e as a derived number which should be greater than 1 and less than (p-1) and (q-1). The primary condition will be that there should be no common factor of (p-1) and (q-1) except 1

Step 3: Public key

The specified pair of numbers **n** and **e** forms the RSA public key and it is made public.

Step 4: Private Key

Private Key **d** is calculated from the numbers p, q and e. The mathematical relationship between the numbers is as follows −

ed = 1 mod (p-1) (q-1)

The above formula is the basic formula for Extended Euclidean Algorithm, which takes p and q as the input parameters.

Encryption Formula

Consider a sender who sends the plain text message to someone whose public key is **(n,e).** To encrypt the plain text message in the given scenario, use the following syntax −

C = Pe mod n

Decryption Formula

The decryption process is very straightforward and includes analytics for calculation in a systematic approach. Considering receiver **C** has the private key **d**, the result modulus will be calculated as −

Plaintext = Cd mod n

Generating RSA keys

The following steps are involved in generating RSA keys −

- Create two large prime numbers namely **p** and **q**. The product of these numbers will be called **n**, where **n= p*q**
- Generate a random number which is relatively prime with **(p-1)** and **(q-1).** Let the number be called as **e**.
- Calculate the modular inverse of e. The calculated inverse will be called as **d**.

Wireshark

Wireshark is a network packet analyzer which tries to capture network packets and tries to display the packet data as detailed as possible. Used for troubleshooting, and monitoring of network and see all details of packet through the network.

It is an open source software. It is also available for many types of OS with GUI environment which provide user friendly interface.

Wireshark lets the user put network interface controllers into promiscuous mode (if supported by the network interface controller), so they can see all the traffic visible on that interface including unicast traffic not sent to that network interface controller's MAC address. However, when capturing with a packet analyzer in promiscuous mode on a port on a network switch, not all traffic through the switch is necessarily sent to the port where the capture is done, so capturing in promiscuous mode is not necessarily sufficient to see all network traffic. Port mirroring or various network taps extend capture to any point on the network. Simple passive taps are extremely resistant to tampering.

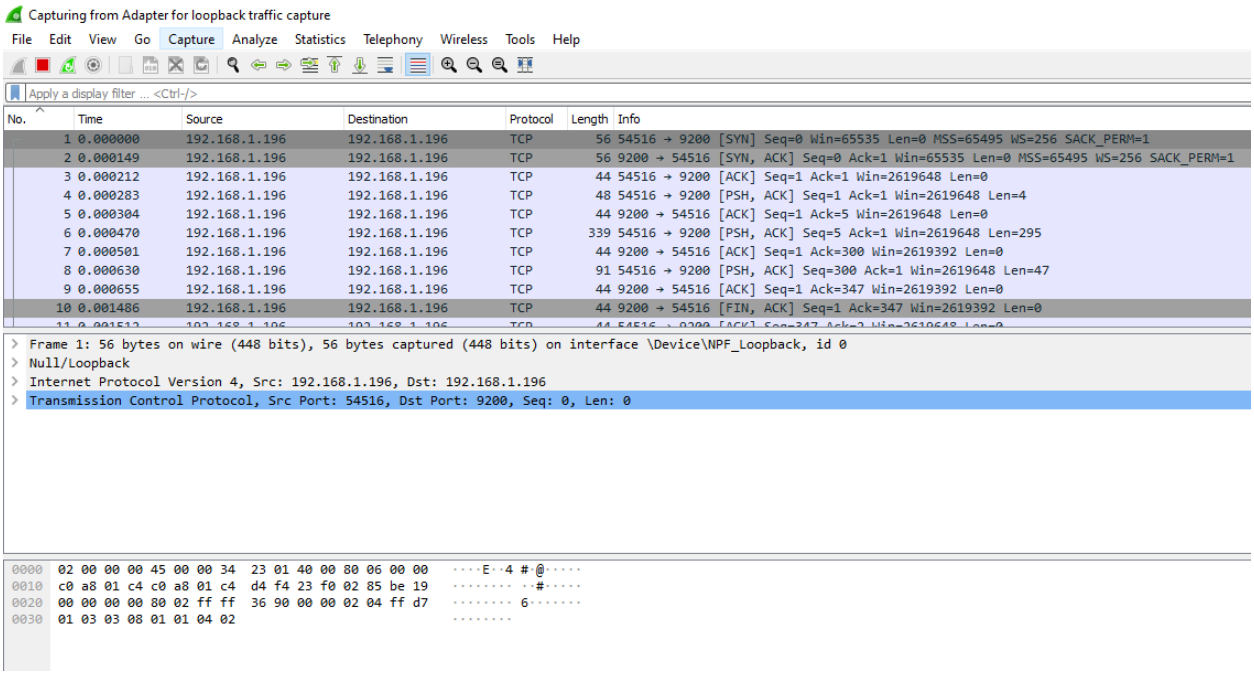
- Used by-
  - network administrators to troubleshoot network problems
  - network security engineers to examine security problems
  - developers to debug protocol implementations

Features:

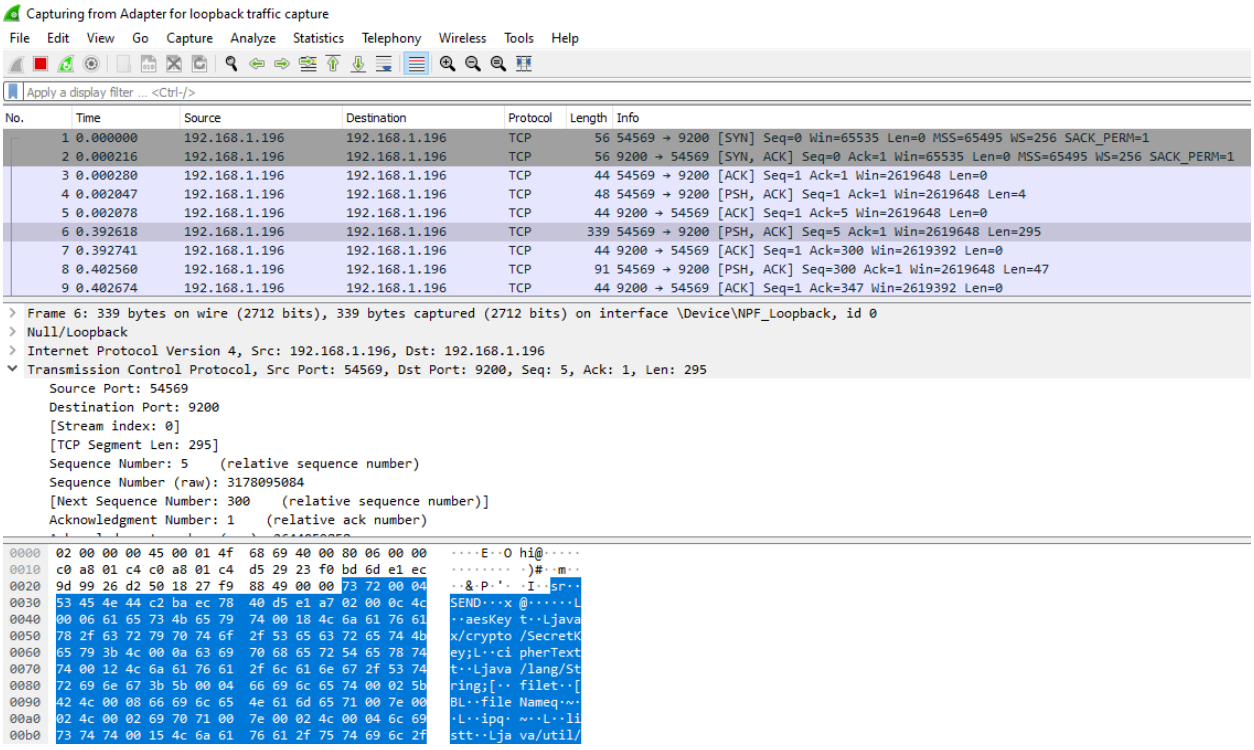
- Available for UNIX and Windows.
- Capture live packet data from a network interface.
- Display packets with very detailed protocol information.
- Open and Save packet data captured.
- Filter packets on many criteria.
- Colorize packet display based on filters.
- Create various statistics.

Observations:

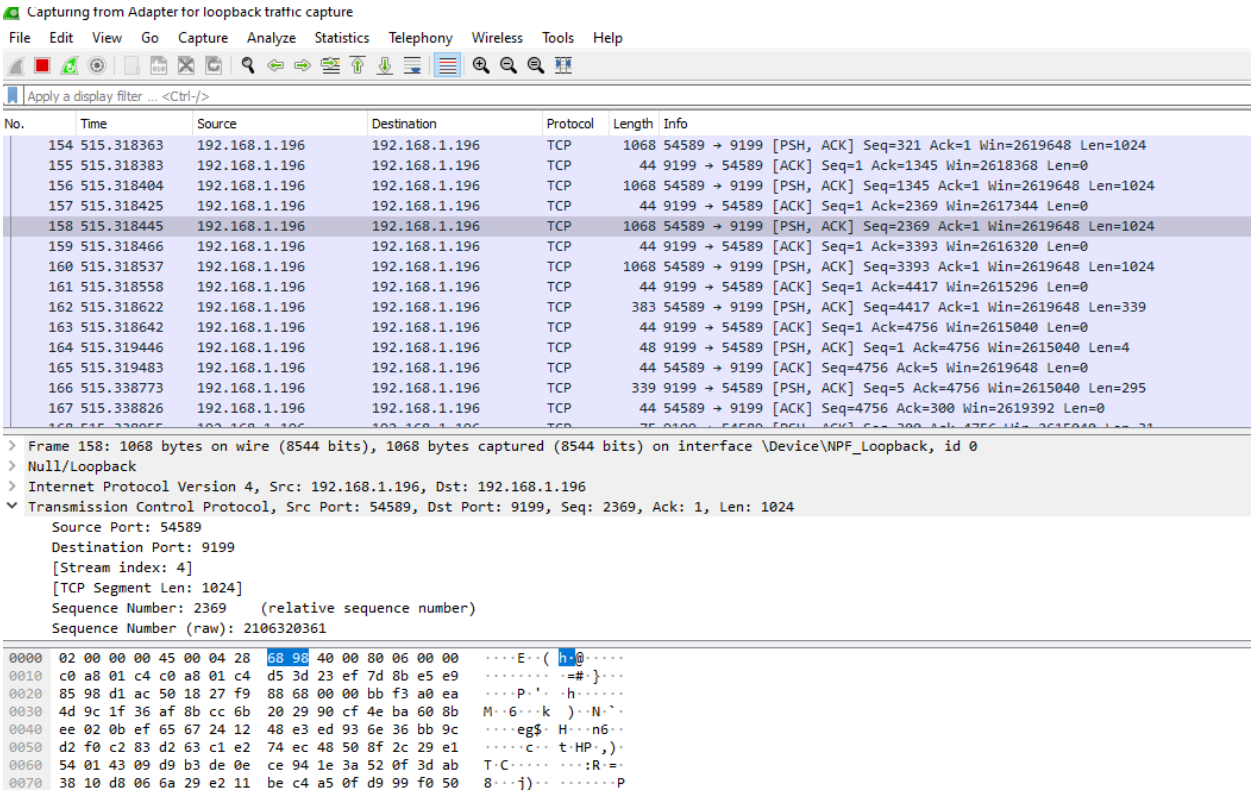
The starting 3 packets simply show 3 way handshake through their sequence and acknowledgement number.



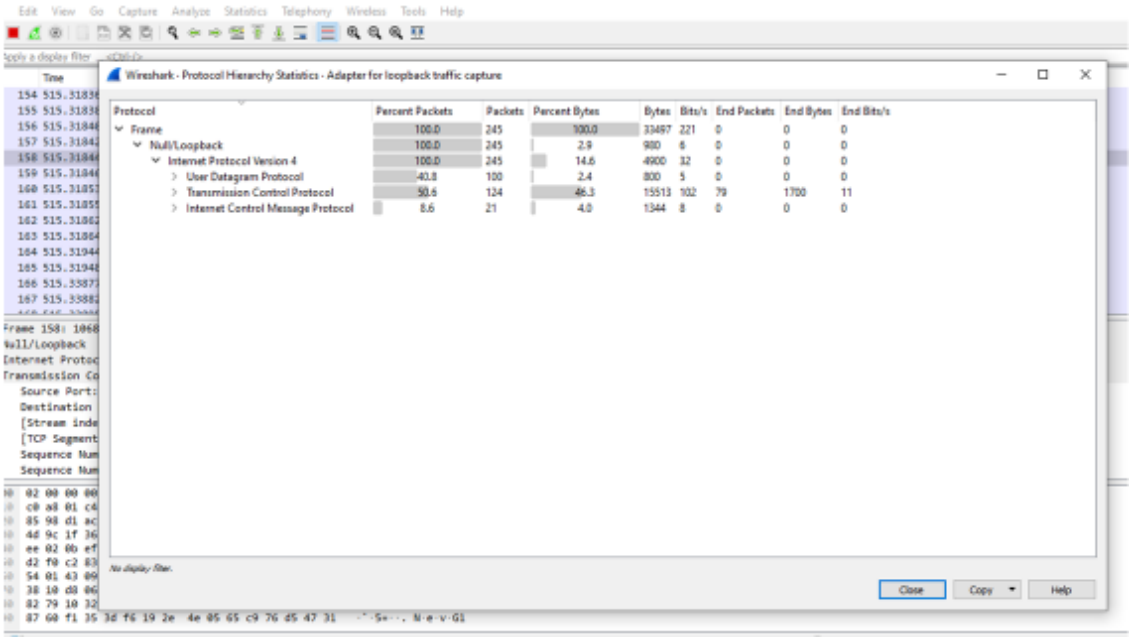
The following image shows the packet details like source, destination, sequence number and many others like length of the packet can be seen by clicking on that packet.



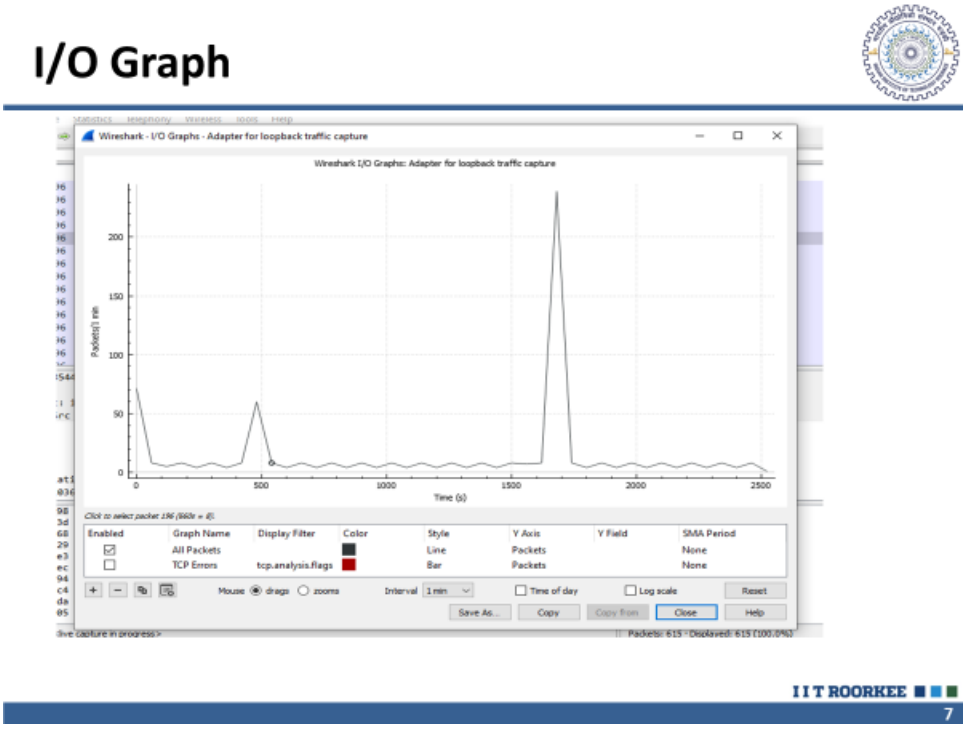
The bottom part of the following image containing the packet data, that data is in encrypted form shows that the data which is transferred is encrypted and secure.



This screenshot shows the different protocols used for transferring the packets and their proportion during the transfer.

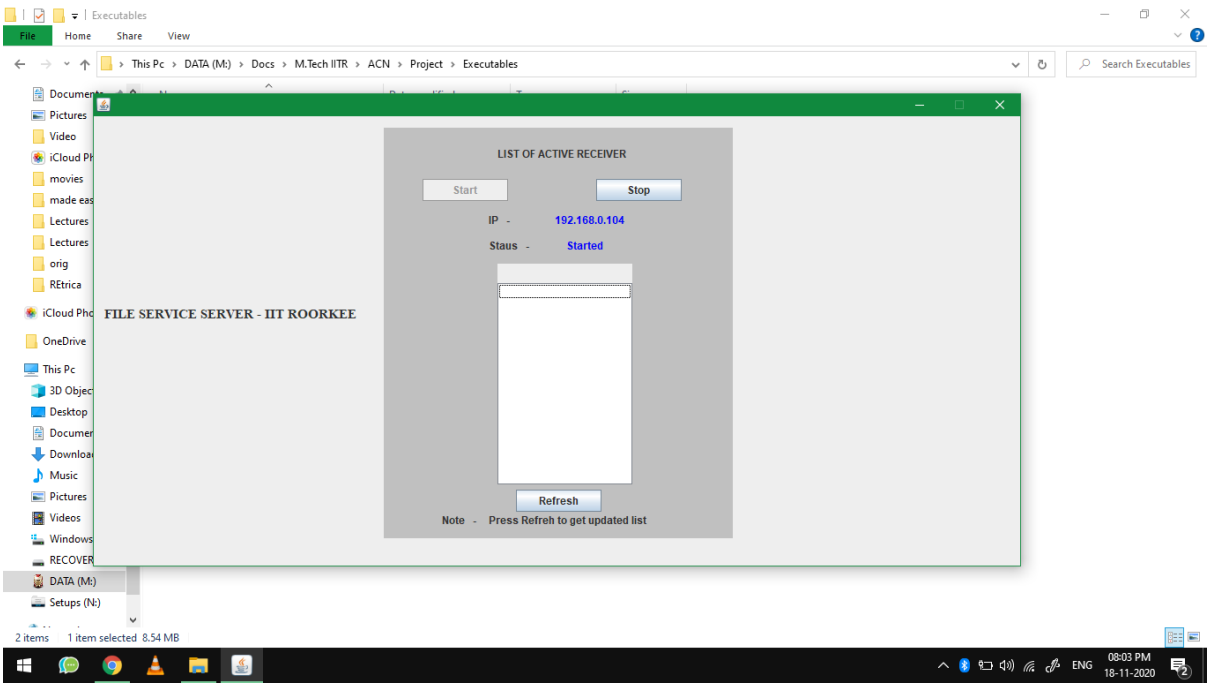


The following image shows the network traffic at various time instances.



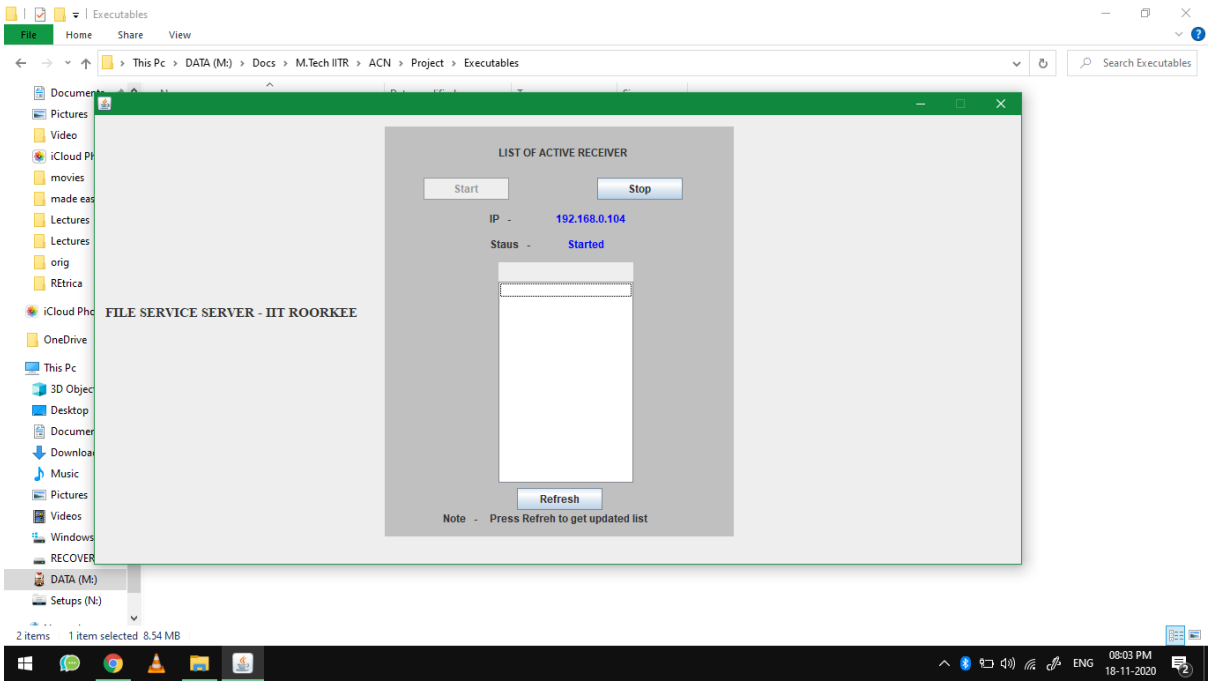
File Service Server :

- Main Motivation of using File Service server Functionality is to find the list of Active Receiver.
- Accept Sender Request and provide them list of active receiver.
- Accept Receiver Request and make them available to all sender who whish to send them.

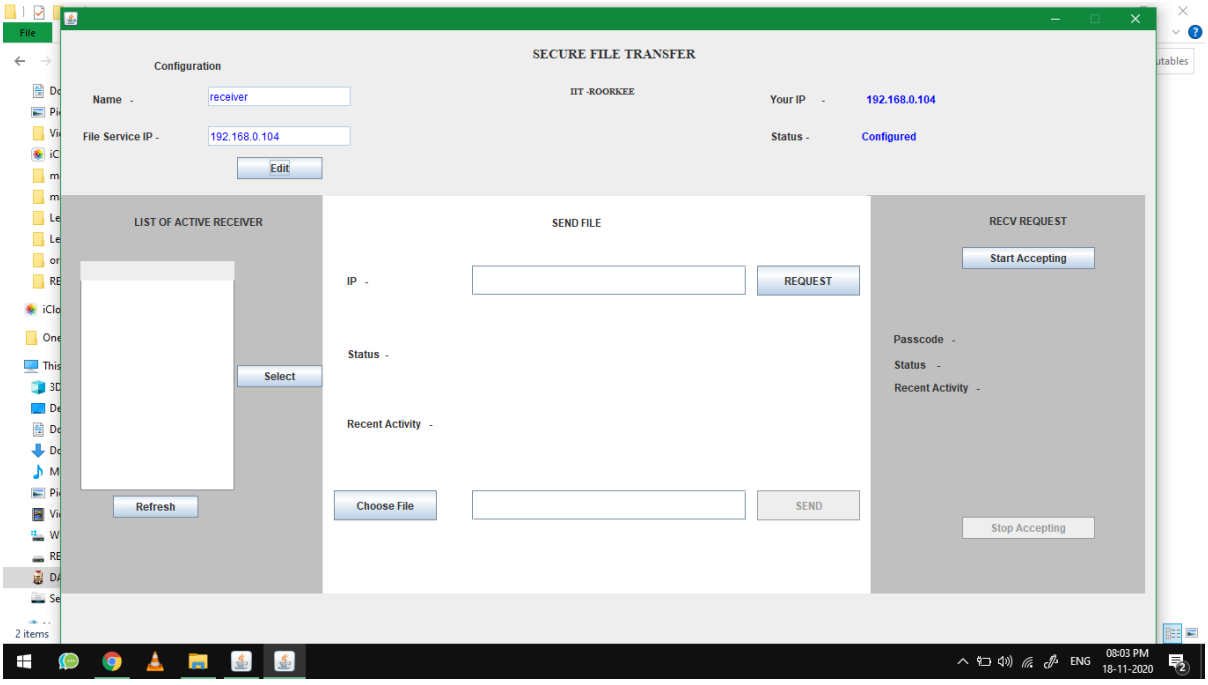


Walkthrough

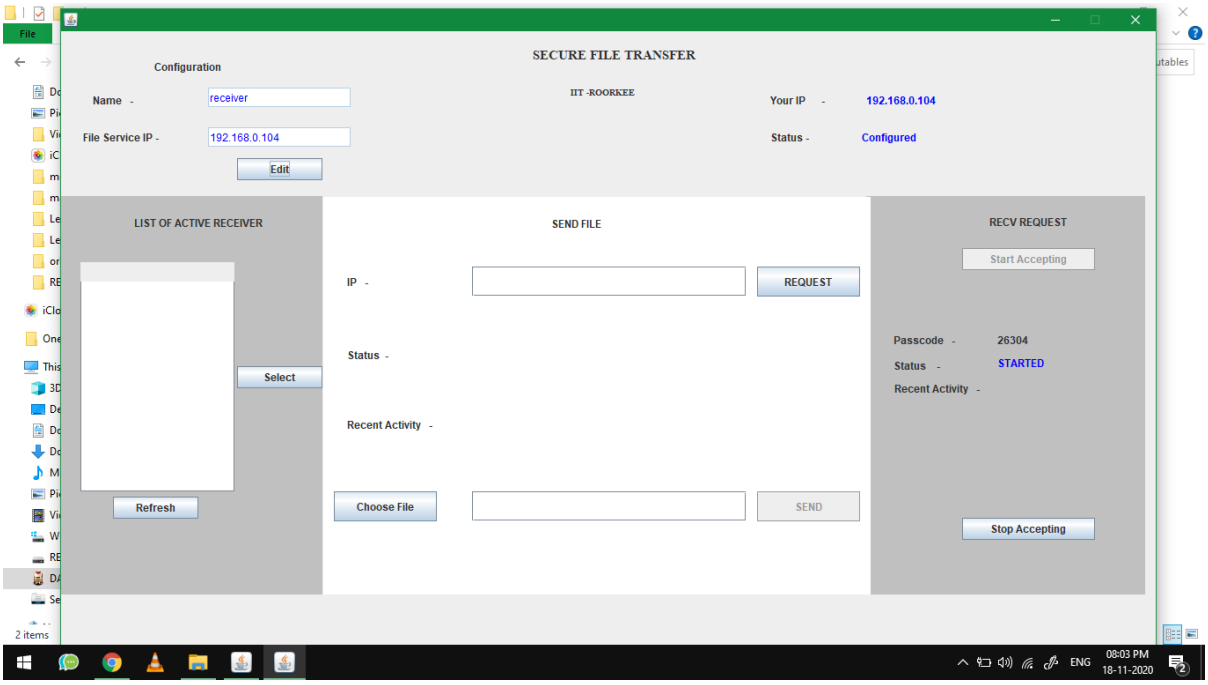
1. First the ‘file service server’ is initiated at any computer in the Lan. (This is only used for monitoring purposes, file transfer can take place without this server as well).



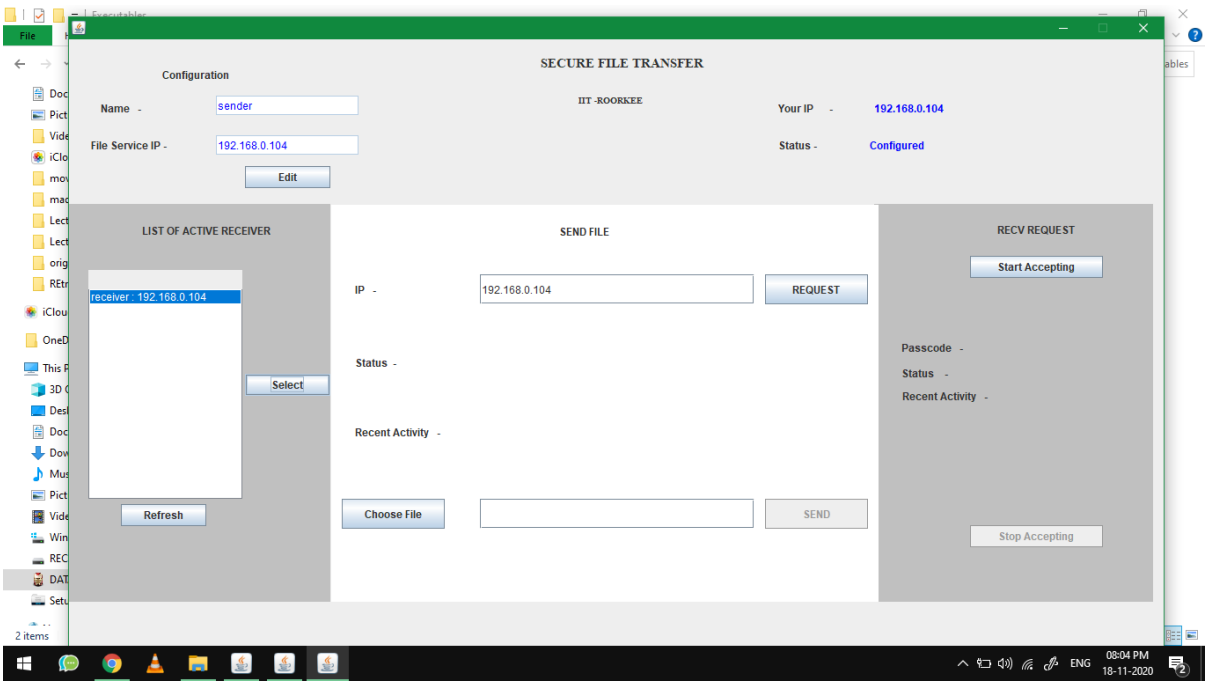
2. The receiver will now configure itself(for monitoring purposes which is optional) .



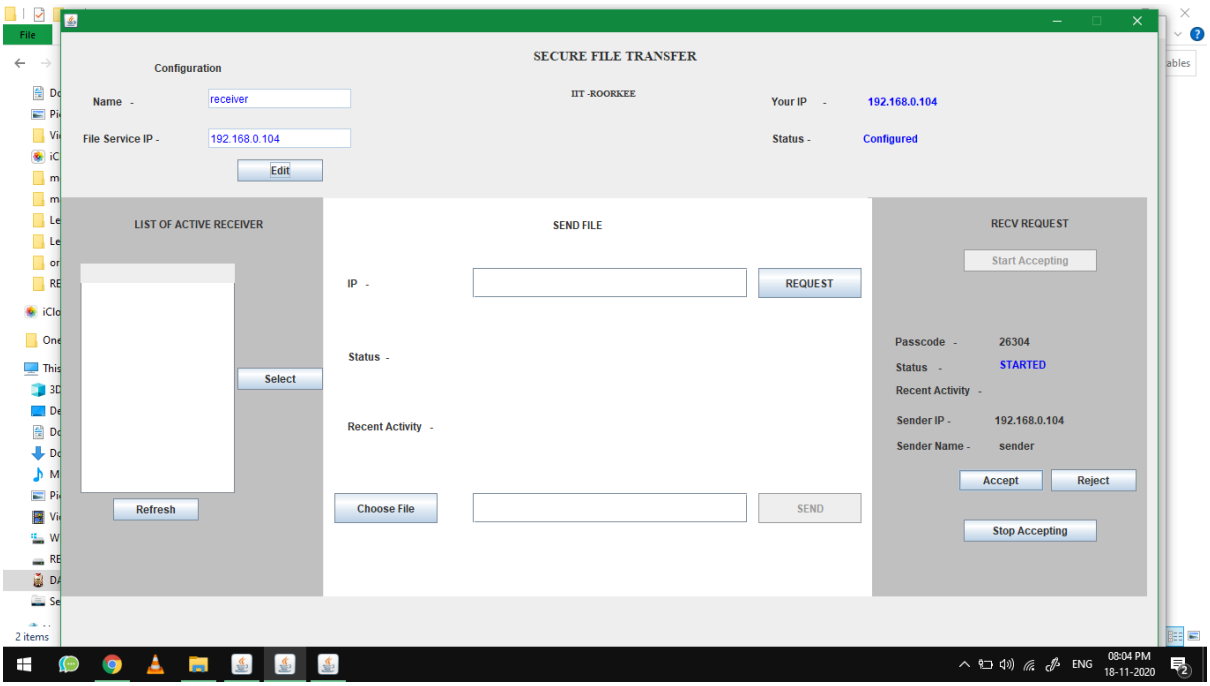
3. Once the configuration is done, the receiver will start receiving by clicking on ‘start accepting’ button.



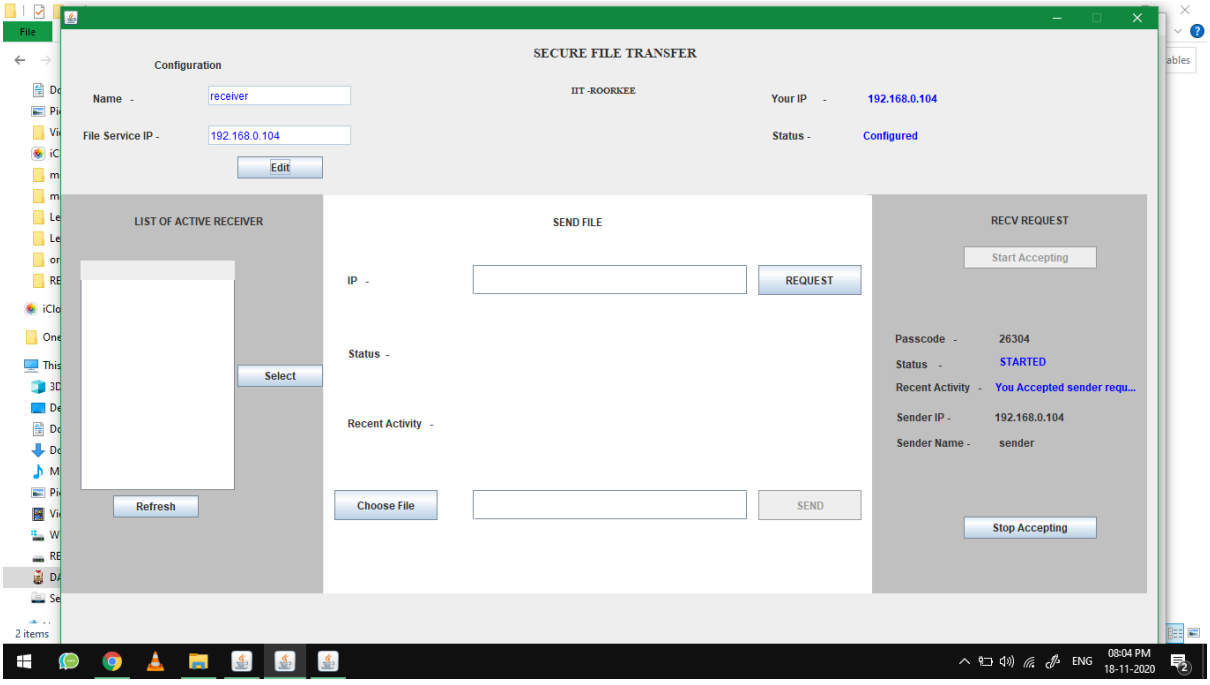
4. Now the sender can also register itself(optional) and select one of receivers in the left panel to transfer file.



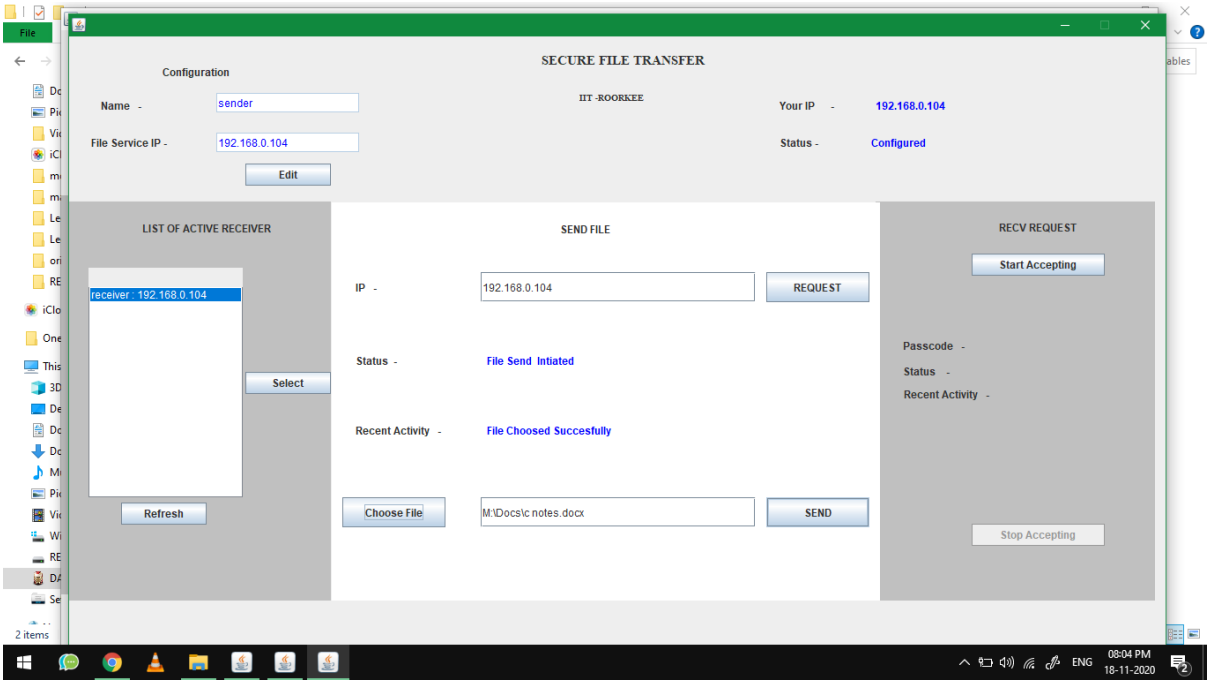
5. Once the sender have clicked on ‘request’ button a request will be sent to receiver as shown below. To accept the request, receiver need to click on ‘accept’ button otherwise receiver can clock on ‘reject’ button.



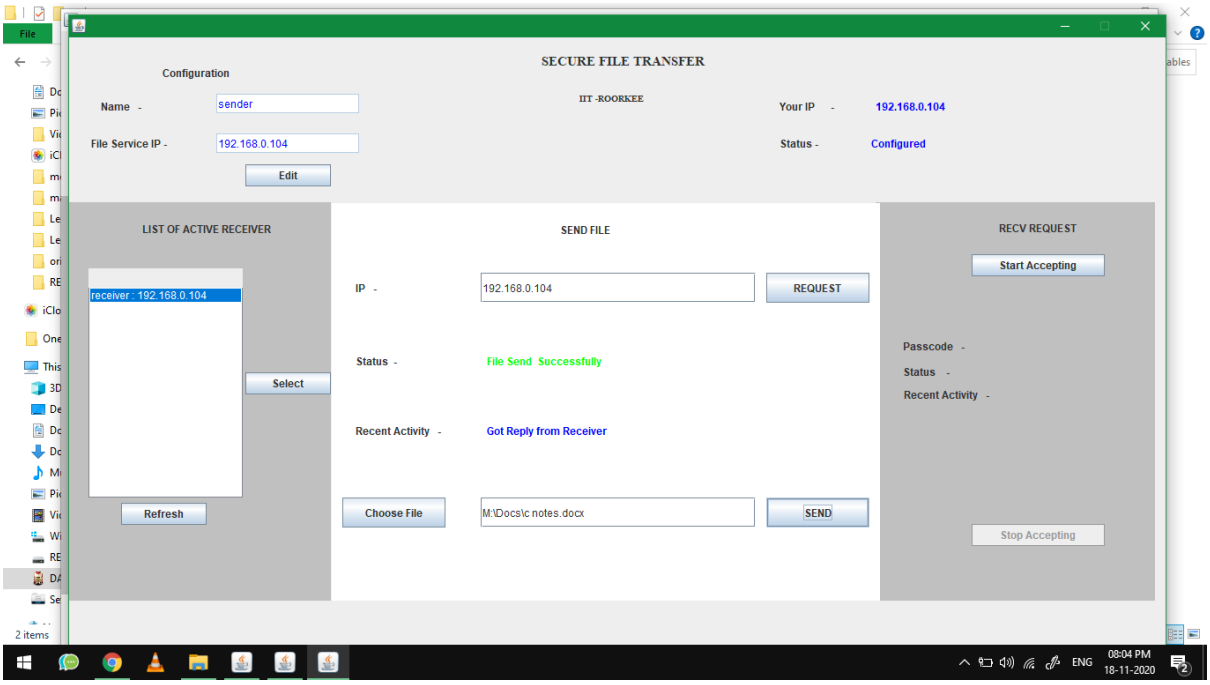
6. If the request is accepted, the status will be updated as “you accepted the sender request”.



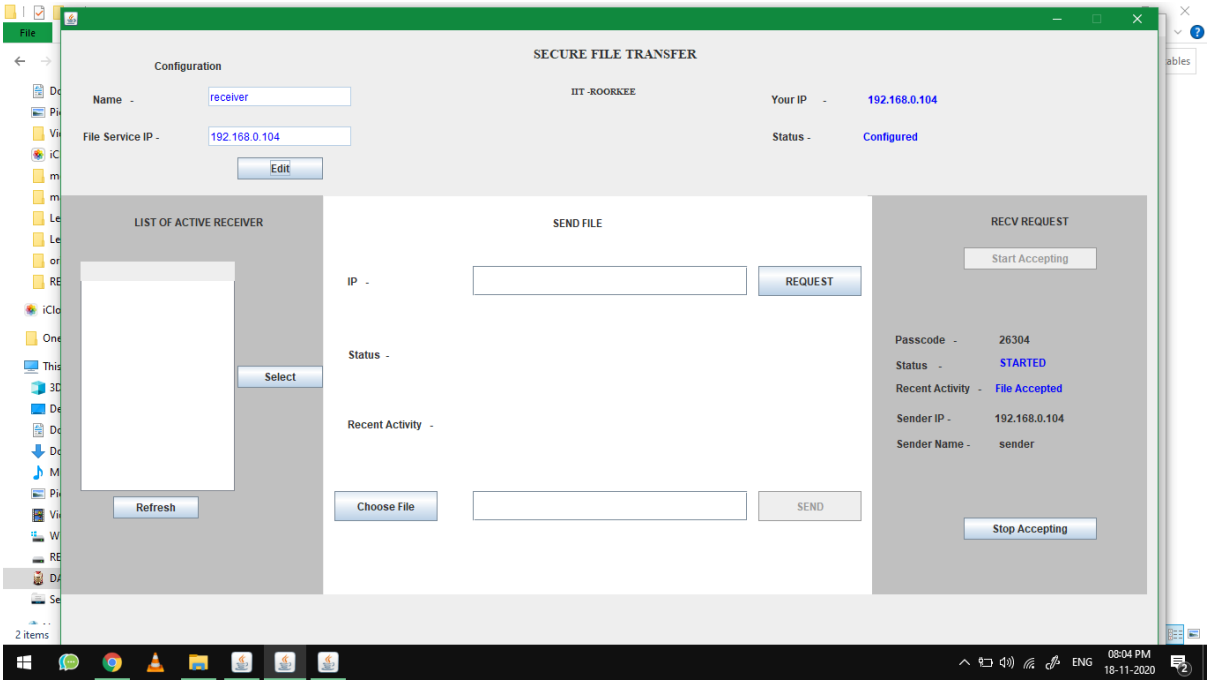
7. At the same time, at senders end, status will be updated as “File Send Initiated”. As shown below.



8. Once the transfer is completed, status will be updated as “File send Successfully”.



9. At receiver's end the status will be updated as "File accepted".



10. At root folder of receivers computer, a directory will be created as RECEIVED/'IP'/'DATE'/ And the received files can be found here(along with encrypted file).

Conclusion

We successfully implemented security protocols for data transfer using socket programming.

The security protocols were a combination of AES and RSA hence our model is a hybrid model which ensures both security and performance.

A central server can be established and it can provide the security using the authentication using the described techniuqes of communication in the campus network where the clients are connected through wired LAN or Ethernet.

References

- https://docs.oracle.com/en/java/javase/15/
- https://www.wireshark.org/docs/