**21CY681– Internet Protocol lab**

**ASSIGNMENT -11**

**Name:** B.Shebu

**Register Number :** CYS22005

**Title:** Application of cryptographical algorithms using socket programming

**Date of Assignment provided:** 1/1/2023

**Aim:** To create a chatbot which implements rsa encryption and sends message from client to server.

WITHOUT RSA IMPLEMENTATION –

SERVER –

import socket

# Create a TCP/IP socket

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# Bind the socket to the port

server\_address = ('localhost', 10000)

print('starting up on {} port {}'.format(\*server\_address))

sock.bind(server\_address

# Listen for incoming connections

sock.listen(1)

while True:

# Wait for a connection

print('waiting for a connection')

connection, client\_address = sock.accept()

try:

print('connection from', client\_address)

# Receive messages from the client and print them

while True:

message = connection.recv(1024).decode()

if message:

print('received message:', message)

else:

print('no data from', client\_address)

break

finally:

# Clean up the connection

connection.close()

CLIENT –

import socket

# Create a TCP/IP socket

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# Connect the socket to the port where the server is listening

server\_address = ('localhost', 10000)

print('connecting to {} port {}'.format(\*server\_address))

sock.connect(server\_address)

try:

while True:

# Read a message from the user and send it to the server

message = input("Enter a message to send to the server (enter 'q' to quit): ")

if message == 'q':

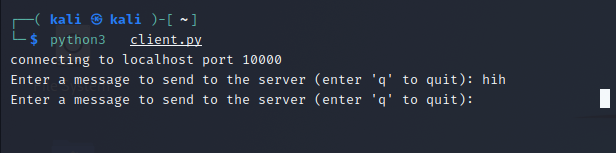
break

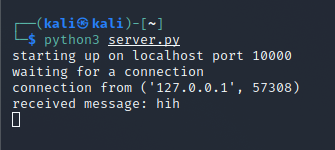
sock.sendall(message.encode())

finally:

sock.close()

SCREENSHOTS –





SERVER.PY

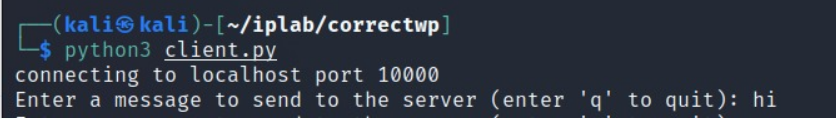
|  |
| --- |
| import socket |
|  | import rsa |
|  |  |
|  | # Generate a new 2048-bit RSA key pair |
|  | (pubkey, privkey) = rsa.newkeys(2048) |
|  |  |
|  | # Create a TCP/IP socket |
|  | sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) |
|  |  |
|  | # Bind the socket to the port |
|  | server\_address = ('localhost', 10000) |
|  | print('starting up on {} port {}'.format(\*server\_address)) |
|  | sock.bind(server\_address) |
|  |  |
|  | # Listen for incoming connections |
|  | sock.listen(1) |
|  |  |
|  | while True: |
|  | # Wait for a connection |
|  | print('waiting for a connection') |
|  | connection, client\_address = sock.accept() |
|  | try: |
|  | print('connection from', client\_address) |
|  |  |
|  | # Receive the client's public key |
|  | client\_pubkey = rsa.PublicKey.load\_pkcs1(connection.recv(1024)) |
|  |  |
|  | # Send the server's public key to the client |
|  | connection.sendall(rsa.PublicKey.save\_pkcs1(pubkey)) |
|  |  |
|  | # Receive encrypted messages from the client and decrypt them using the server's private key |
|  | while True: |
|  | encrypted\_message = connection.recv(1024) |
|  | if encrypted\_message: |
|  | message = rsa.decrypt(encrypted\_message, privkey).decode() |
|  | print('received message:', message) |
|  | else: |
|  | print('no data from', client\_address) |
|  | break |
|  | finally: |
|  | # Clean up the connection |
|  | connection.close() |

CLIENT.PY

|  |  |
| --- | --- |
|  | import rsa  import socket |
|  |  |
|  | # Generate a new 2048-bit RSA key pair |
|  | (pubkey, privkey) = rsa.newkeys(2048) |
|  |  |
|  | # Create a TCP/IP socket |
|  | sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) |
|  |  |
|  | # Connect the socket to the port where the server is listening |
|  | server\_address = ('localhost', 10000) |
|  | print('connecting to {} port {}'.format(\*server\_address)) |
|  | sock.connect(server\_address) |
|  |  |
|  | try: |
|  | # Send the client's public key to the server |
|  | sock.sendall(rsa.PublicKey.save\_pkcs1(pubkey)) |
|  |  |
|  | # Receive the server's public key |
|  | server\_pubkey = rsa.PublicKey.load\_pkcs1(sock.recv(1024)) |
|  |  |
|  | while True: |
|  | # Read a message from the user and send it to the server |
|  | message = input("Enter a message to send to the server (enter 'q' to quit): ") |
|  | if message == 'q': |
|  | break |
|  | encrypted\_message = rsa.encrypt(message.encode(), server\_pubkey) |
|  | sock.sendall(encrypted\_message) |
|  | finally: |
|  | sock.close() |

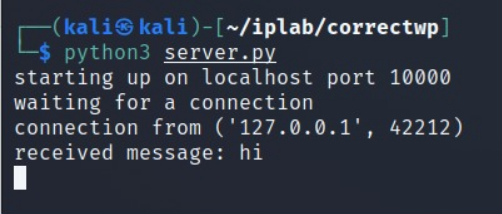
SCREENSHOTS –

CLIENT

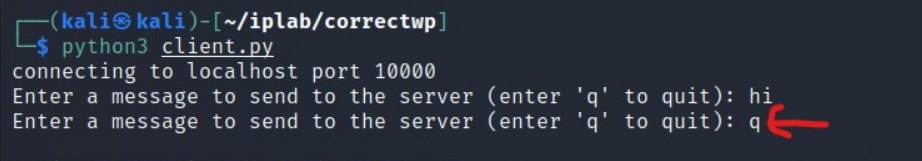


In the server we send a message called “ hi “

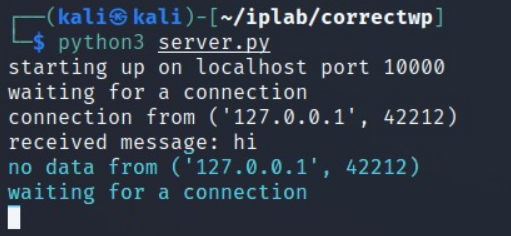
SERVER



Here we get the message from the client.

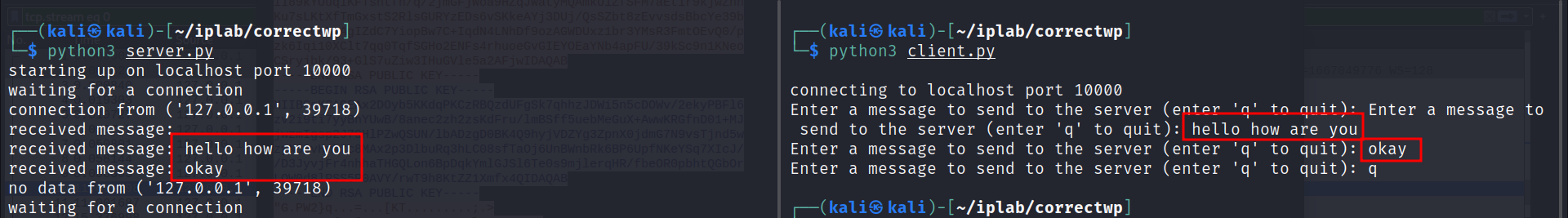


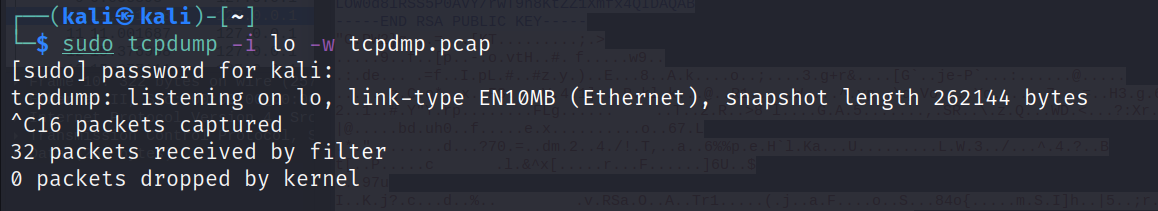
We can disconnect the connection to the server by sending the message “q”



Since the server sent q message the connection is terminated and the server is again waiting for any new connection .

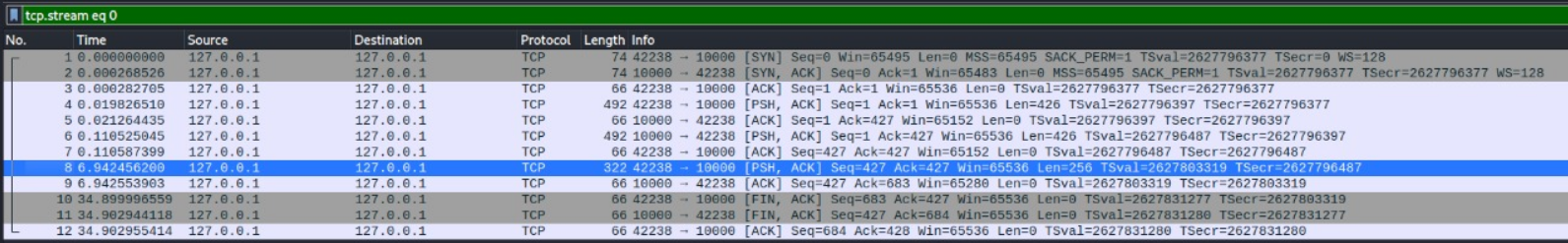
TCPDUMP –

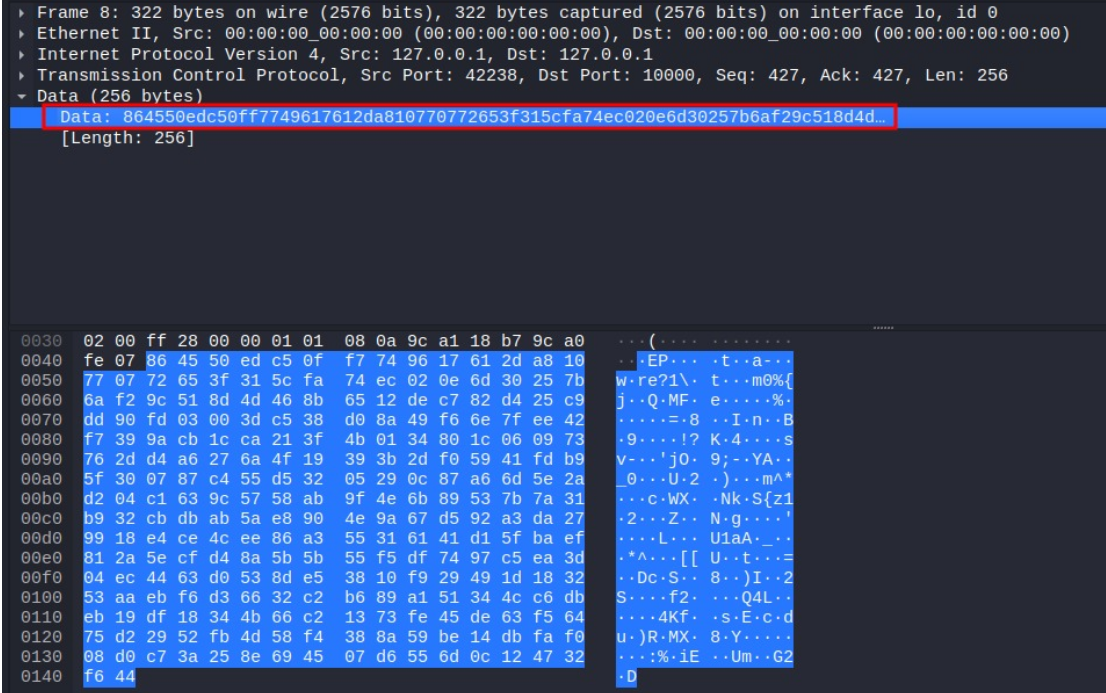


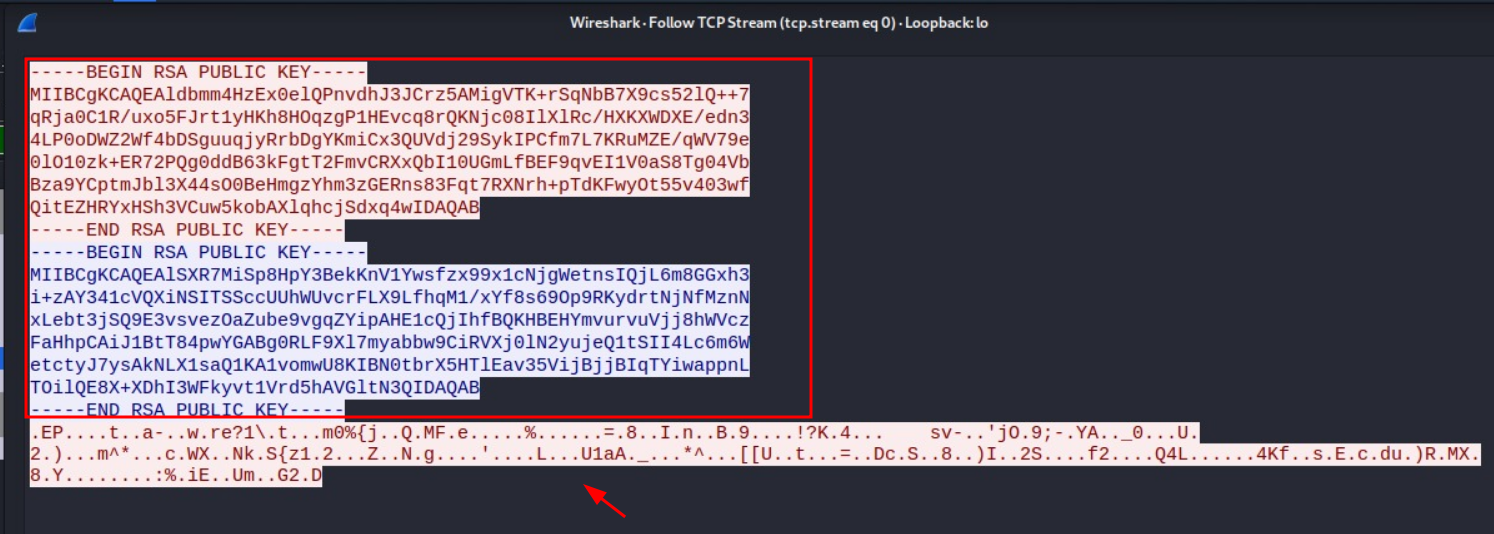


Now we captured the traffic using tcpdump.

WIRESHARK –







In the above screenshot we can see that we captured a packet which has the public key and some encrypted data .