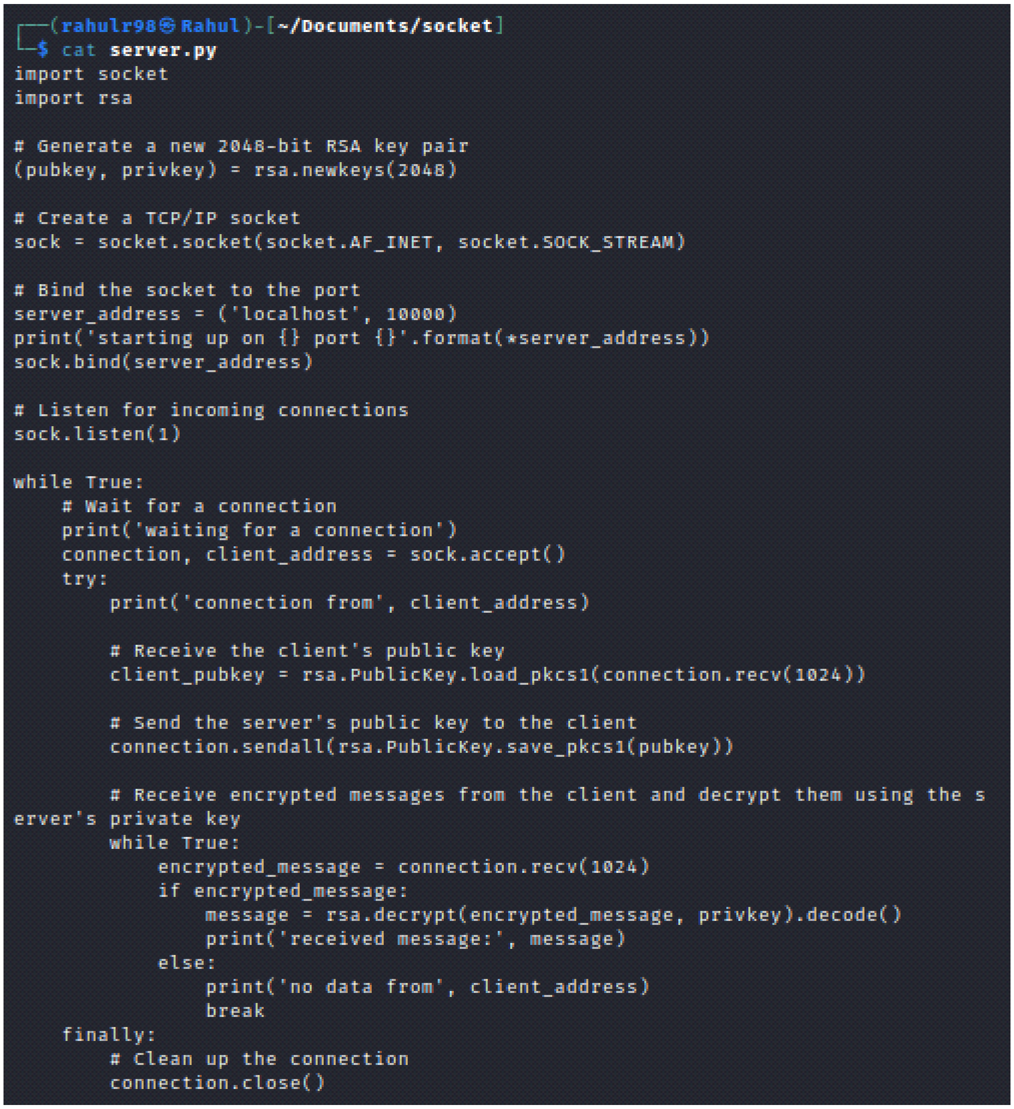
ASSIGNMENT
INTERNET PROTOCOL LAB

NAME: Rahul Raj
REG NO: CYS22011
DATE OF ASSIGNMENT PROVIDED: 05/1/2023
TITLE: Application of Cryptographical Algorithmd uding Socket programming.

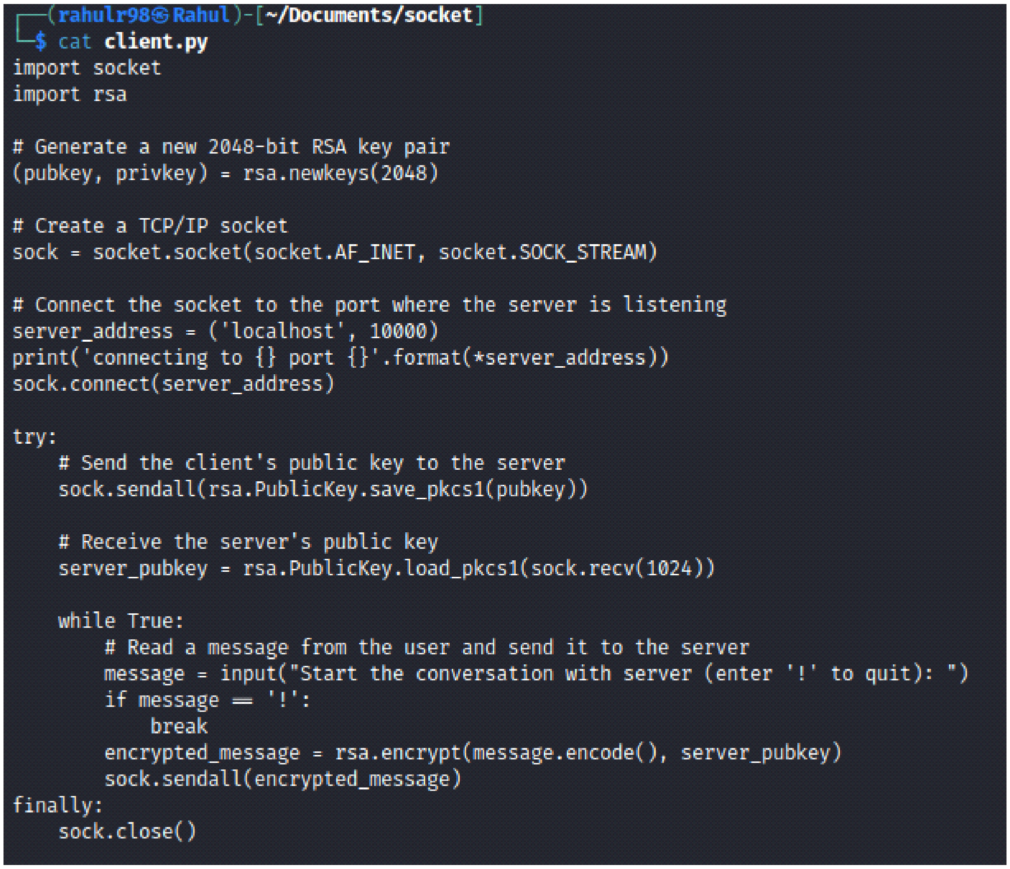
Establish a Client-Server Secure communication protocol-

- Python program for Server establishment with RSA encryption.



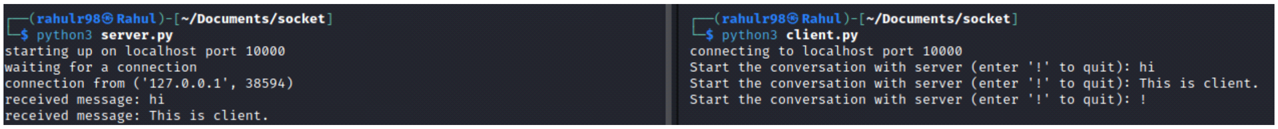
```
┌──(rahulr98㉿Rahul)-[~/Documents/socket]
└─$ cat server.py
import socket
import rsa

# 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 s
erver'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()
```

- Python program for Client establishment with RSA encryption.



```
┌──(rahulr98㉿Rahul)-[~/Documents/socket]
└─$ cat client.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)

# 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("Start the conversation with server (enter '!' to quit): ")
        if message == '!':
            break
        encrypted_message = rsa.encrypt(message.encode(), server_pubkey)
        sock.sendall(encrypted_message)
finally:
    sock.close()
```

- Now we have to run these two programs in different terminal to establish the connection.



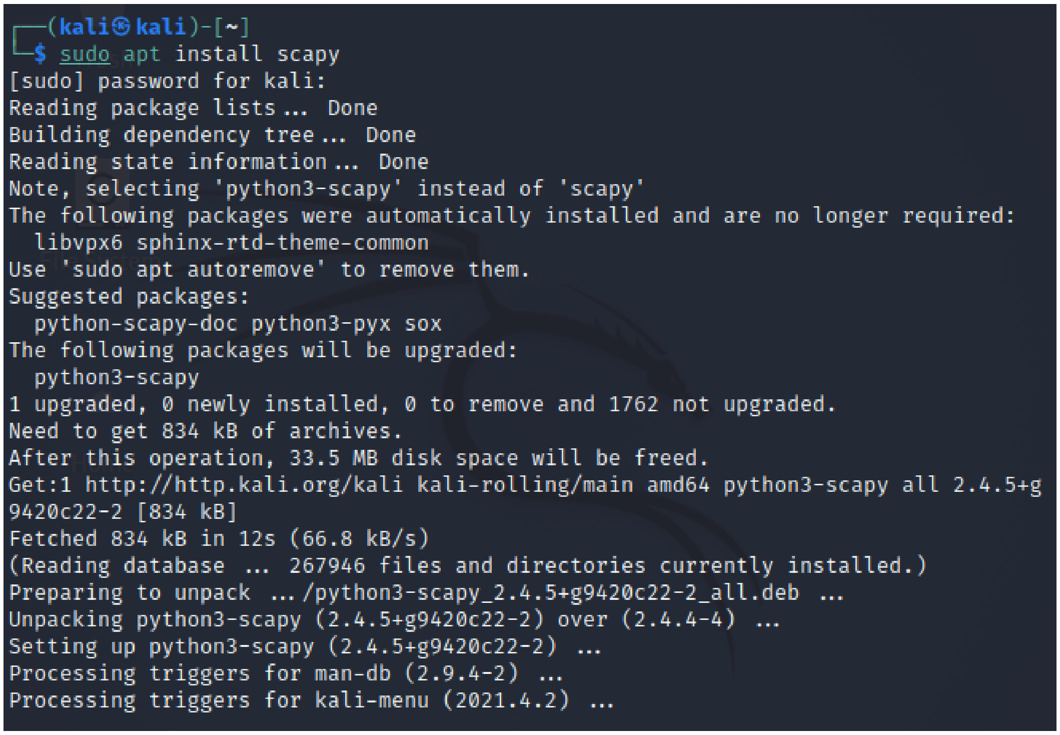
```
┌──(rahulr98㉿Rahul)-[~/Documents/socket]
└─$ python3 server.py
starting up on localhost port 10000
waiting for a connection
connection from ('127.0.0.1', 38594)
received message: hi
received message: This is client.
```

```
┌──(rahulr98㉿Rahul)-[~/Documents/socket]
└─$ python3 client.py
connecting to localhost port 10000
Start the conversation with server (enter '!' to quit): hi
Start the conversation with server (enter '!' to quit): This is client.
Start the conversation with server (enter '!' to quit): !
```

Here, we can see that the Server and Client are connected successfully and exchanged messages successfully.

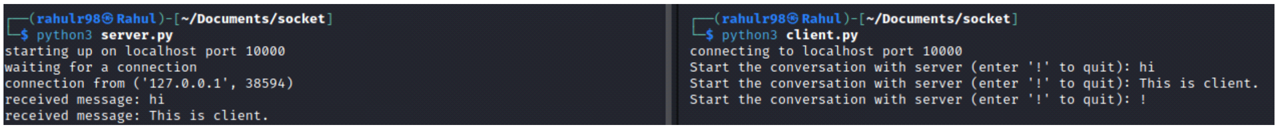
Intalling SCAPY for sniffing and Capturing Packets-

- Installing Scapy.



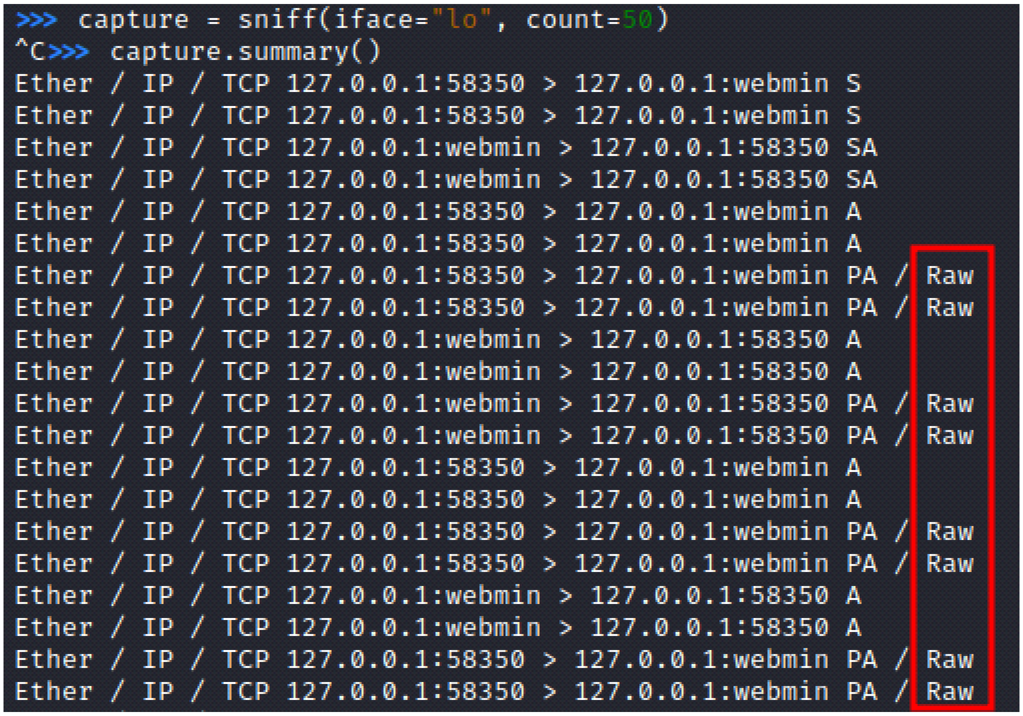
```
┌──(kali㉿kali)-[~]
└─$ sudo apt install scapy
[sudo] password for kali:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'python3-scapy' instead of 'scapy'
The following packages were automatically installed and are no longer required:
  libvpx6 sphinx-rtd-theme-common
Use 'sudo apt autoremove' to remove them.
Suggested packages:
  python-scapy-doc python3-pyx sox
The following packages will be upgraded:
  python3-scapy
1 upgraded, 0 newly installed, 0 to remove and 1762 not upgraded.
Need to get 834 kB of archives.
After this operation, 33.5 MB disk space will be freed.
Get:1 http://http.kali.org/kali kali-rolling/main amd64 python3-scapy all 2.4.5+g
9420c22-2 [834 kB]
Fetched 834 kB in 12s (66.8 kB/s)
(Reading database ... 267946 files and directories currently installed.)
Preparing to unpack .../python3-scapy_2.4.5+g9420c22-2_all.deb ...
Unpacking python3-scapy (2.4.5+g9420c22-2) over (2.4.4-4) ...
Setting up python3-scapy (2.4.5+g9420c22-2) ...
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for kali-menu (2021.4.2) ...
```

- Now run the Server-Client program and start capturing the packets.



```
┌──(rahulr98㉿Rahul)-[~/Documents/socket]
└─$ python3 server.py
starting up on localhost port 10000
waiting for a connection
connection from ('127.0.0.1', 38594)
received message: hi
received message: This is client.
```

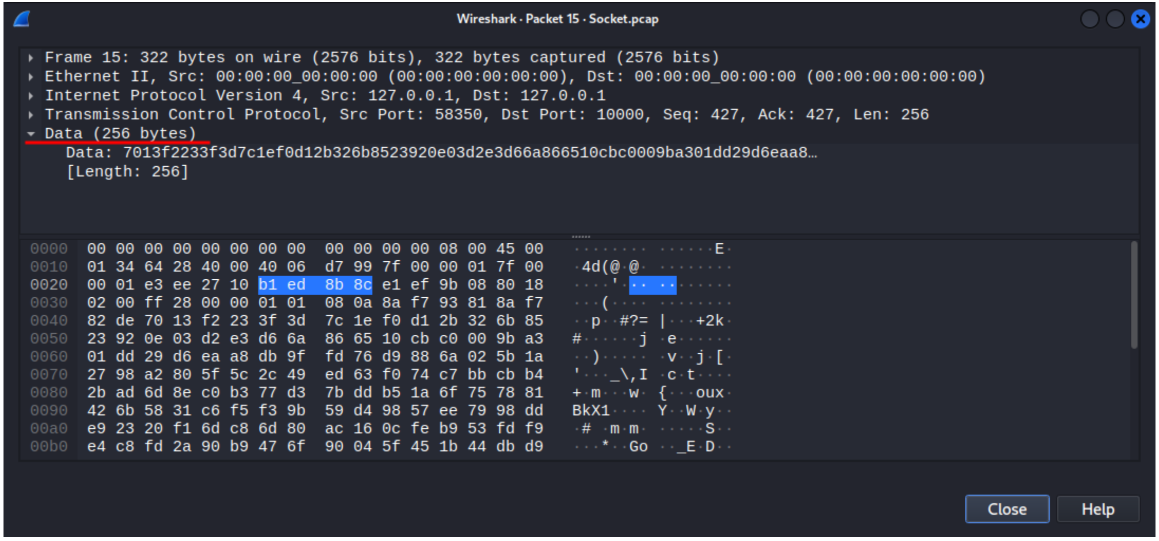
```
┌──(rahulr98㉿Rahul)-[~/Documents/socket]
└─$ python3 client.py
connecting to localhost port 10000
Start the conversation with server (enter '!' to quit): hi
Start the conversation with server (enter '!' to quit): This is client.
Start the conversation with server (enter '!' to quit): !
```



```
>>> capture = sniff(iface="lo", count=50)
^C>>> capture.summary()
Ether / IP / TCP 127.0.0.1:58350 > 127.0.0.1:webmin S
Ether / IP / TCP 127.0.0.1:58350 > 127.0.0.1:webmin S
Ether / IP / TCP 127.0.0.1:webmin > 127.0.0.1:58350 SA
Ether / IP / TCP 127.0.0.1:webmin > 127.0.0.1:58350 SA
Ether / IP / TCP 127.0.0.1:58350 > 127.0.0.1:webmin A
Ether / IP / TCP 127.0.0.1:58350 > 127.0.0.1:webmin A
Ether / IP / TCP 127.0.0.1:58350 > 127.0.0.1:webmin PA / Raw
Ether / IP / TCP 127.0.0.1:58350 > 127.0.0.1:webmin PA / Raw
Ether / IP / TCP 127.0.0.1:webmin > 127.0.0.1:58350 A
Ether / IP / TCP 127.0.0.1:webmin > 127.0.0.1:58350 A
Ether / IP / TCP 127.0.0.1:webmin > 127.0.0.1:58350 PA / Raw
Ether / IP / TCP 127.0.0.1:webmin > 127.0.0.1:58350 PA / Raw
Ether / IP / TCP 127.0.0.1:58350 > 127.0.0.1:webmin A
Ether / IP / TCP 127.0.0.1:58350 > 127.0.0.1:webmin A
Ether / IP / TCP 127.0.0.1:58350 > 127.0.0.1:webmin PA / Raw
Ether / IP / TCP 127.0.0.1:58350 > 127.0.0.1:webmin PA / Raw
Ether / IP / TCP 127.0.0.1:webmin > 127.0.0.1:58350 A
Ether / IP / TCP 127.0.0.1:webmin > 127.0.0.1:58350 A
Ether / IP / TCP 127.0.0.1:58350 > 127.0.0.1:webmin PA / Raw
Ether / IP / TCP 127.0.0.1:58350 > 127.0.0.1:webmin PA / Raw
```

Here, we captured packets using ***capture = sniff(iface="lo",count=50)*** command. Here the Raw showing packets contains data.

Now we are saving the pcap and open it in wireshark.



Wireshark · Packet 15 · Socket.pcap

```
▸ Frame 15: 322 bytes on wire (2576 bits), 322 bytes captured (2576 bits)
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 58350, Dst Port: 10000, Seq: 427, Ack: 427, Len: 256
▾ Data (256 bytes)
    Data: 7013f2233f3d7c1ef0d12b326b8523920e03d2e3d66a866510cbc0009ba301dd29d6eaa8…
    [Length: 256]

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  01 34 64 28 40 00 40 06  d7 99 7f 00 00 01 7f 00   ·4d(@·@· ········
0020  00 01 e3 ee 27 10 b1 ed  8b 8c e1 ef 9b 08 80 18   ····'··· ········
0030  02 00 ff 28 00 00 01 01  08 0a 8a f7 93 81 8a f7   ···(···· ········
0040  82 de 70 13 f2 23 3f 3d  7c 1e f0 d1 2b 32 6b 85   ··p··#?= |···+2k·
0050  23 92 0e 03 d2 e3 d6 6a  86 65 10 cb c0 00 9b a3   #······j ·e······
0060  01 dd 29 d6 ea a8 db 9f  fd 76 d9 88 6a 02 5b 1a   ··)····· ·v··j·[·
0070  27 98 a2 80 5f 5c 2c 49  ed 63 f0 74 c7 bb cb b4   '···_\,I ·c·t····
0080  2b ad 6d 8e c0 b3 77 d3  7b dd b5 1a 6f 75 78 81   +·m···w· {···oux·
0090  42 6b 58 31 c6 f5 f3 9b  59 d4 98 57 ee 79 98 dd   BkX1···· Y··W·y··
00a0  e9 23 20 f1 6d c8 6d 80  ac 16 0c fe b9 53 fd f9   ·# ·m·m· ·····S··
00b0  e4 c8 fd 2a 90 b9 47 6f  90 04 5f 45 1b 44 db d9   ···*··Go ··_E·D··
```