21CY681– Internet Protocol lab

ASSIGNMENT -11

Name: Ramya Ajay

Register Number : CYS22004

Title: Application of cryptographical algorithms using socket programming

Date of Assignment provided: 5/1/2023

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

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()
```

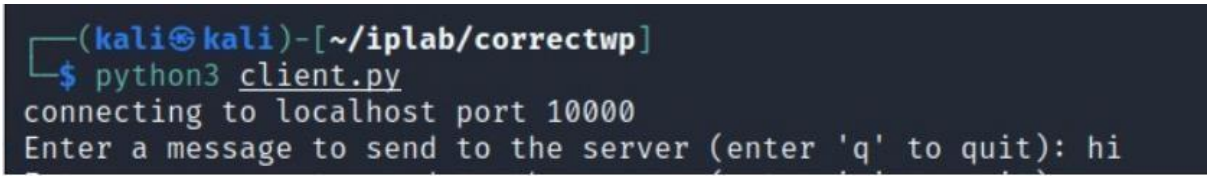
CLIENT.PY

```
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()
```

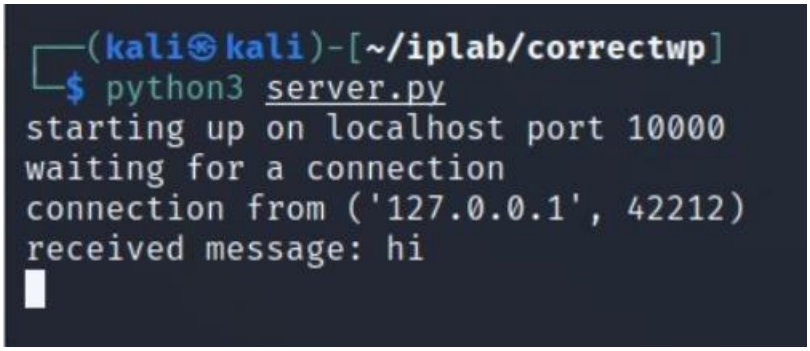
SCREENSHOTS –

CLIENT



In the server we send a message called “ hi “

SERVER



Here we get the message from the client.

```
┌──(kali㉿kali)-[~/iplab/correctwp]
└─$ python3 client.py
connecting to localhost port 10000
Enter a message to send to the server (enter 'q' to quit): hi
Enter a message to send to the server (enter 'q' to quit): q
```

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

```
┌──(kali㉿kali)-[~/iplab/correctwp]
└─$ python3 server.py
starting up on localhost port 10000
waiting for a connection
connection from ('127.0.0.1', 42212)
received message: hi
no data from ('127.0.0.1', 42212)
waiting for a connection
```

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

WIRESHARK –

tcp.stream eq 0

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	74	42238 → 10000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=2627796377 TSecr=0 WS=128
2	0.000268526	127.0.0.1	127.0.0.1	TCP	74	10000 → 42238 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=2627796377 TSecr=2627796377 WS=128
3	0.000282705	127.0.0.1	127.0.0.1	TCP	66	42238 → 10000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2627796377 TSecr=2627796377
4	0.019826510	127.0.0.1	127.0.0.1	TCP	492	42238 → 10000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=426 TSval=2627796397 TSecr=2627796377
5	0.021264435	127.0.0.1	127.0.0.1	TCP	66	10000 → 42238 [ACK] Seq=1 Ack=427 Win=65152 Len=0 TSval=2627796397 TSecr=2627796397
6	0.110525045	127.0.0.1	127.0.0.1	TCP	492	10000 → 42238 [PSH, ACK] Seq=1 Ack=427 Win=65536 Len=426 TSval=2627796487 TSecr=2627796397
7	0.110587399	127.0.0.1	127.0.0.1	TCP	66	42238 → 10000 [ACK] Seq=427 Ack=427 Win=65152 Len=0 TSval=2627796487 TSecr=2627796487
8	6.942456200	127.0.0.1	127.0.0.1	TCP	322	42238 → 10000 [PSH, ACK] Seq=427 Ack=427 Win=65536 Len=256 TSval=2627803319 TSecr=2627796487
9	6.942553903	127.0.0.1	127.0.0.1	TCP	66	10000 → 42238 [ACK] Seq=427 Ack=683 Win=65280 Len=0 TSval=2627803319 TSecr=2627803319
10	34.899996559	127.0.0.1	127.0.0.1	TCP	66	42238 → 10000 [FIN, ACK] Seq=683 Ack=427 Win=65536 Len=0 TSval=2627831277 TSecr=2627803319
11	34.902944118	127.0.0.1	127.0.0.1	TCP	66	10000 → 42238 [FIN, ACK] Seq=427 Ack=684 Win=65536 Len=0 TSval=2627831280 TSecr=2627831277
12	34.902955414	127.0.0.1	127.0.0.1	TCP	66	42238 → 10000 [ACK] Seq=684 Ack=428 Win=65536 Len=0 TSval=2627831280 TSecr=2627831280

```
▸ Frame 8: 322 bytes on wire (2576 bits), 322 bytes captured (2576 bits) on interface lo, id 0
▸ 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: 42238, Dst Port: 10000, Seq: 427, Ack: 427, Len: 256
▾ Data (256 bytes)
    Data: 864550edc50ff7749617612da810770772653f315cfa74ec020e6d30257b6af29c518d4d…
    [Length: 256]

0030  02 00 ff 28 00 00 01 01  08 0a 9c a1 18 b7 9c a0   ···(···· ........
0040  fe 07 86 45 50 ed c5 0f  f7 74 96 17 61 2d a8 10   ··EP··· ·t··a-··
0050  77 07 72 65 3f 31 5c fa  74 ec 02 0e 6d 30 25 7b   w·re?1\· t···m0%{
0060  6a f2 9c 51 8d 4d 46 8b  65 12 de c7 82 d4 25 c9   j··Q·MF· e·····%·
0070  dd 90 fd 03 00 3d c5 38  d0 8a 49 f6 6e 7f ee 42   ·····=·8 ··I·n··B
0080  f7 39 9a cb 1c ca 21 3f  4b 01 34 80 1c 06 09 73   ·9····!? K·4····s
0090  76 2d d4 a6 27 6a 4f 19  39 3b 2d f0 59 41 fd b9   v-··'jO· 9;-·YA··
00a0  5f 30 07 87 c4 55 d5 32  05 29 0c 87 a6 6d 5e 2a   _0···U·2 ·)···m^*
00b0  d2 04 c1 63 9c 57 58 ab  9f 4e 6b 89 53 7b 7a 31   ···c·WX· ·Nk·S{z1
00c0  b9 32 cb db ab 5a e8 90  4e 9a 67 d5 92 a3 da 27   ·2···Z·· N·g····'
00d0  99 18 e4 ce 4c ee 86 a3  55 31 61 41 d1 5f ba ef   ····L··· U1aA·_··
00e0  81 2a 5e cf d4 8a 5b 5b  55 f5 df 74 97 c5 ea 3d   ·*^···[[ U··t···=
00f0  04 ec 44 63 d0 53 8d e5  38 10 f9 29 49 1d 18 32   ··Dc·S·· 8··)I··2
0100  53 aa eb f6 d3 66 32 c2  b6 89 a1 51 34 4c c6 db   S····f2· ···Q4L··
0110  eb 19 df 18 34 4b 66 c2  13 73 fe 45 de 63 f5 64   ····4Kf· ·s·E·c·d
0120  75 d2 29 52 fb 4d 58 f4  38 8a 59 be 14 db fa f0   u·)R·MX· 8·Y·····
0130  08 d0 c7 3a 25 8e 69 45  07 d6 55 6d 0c 12 47 32   ···:%·iE ··Um··G2
0140  f6 44                                              ·D
```

```
Wireshark · Follow TCP Stream (tcp.stream eq 0) · Loopback: lo

-----BEGIN RSA PUBLIC KEY-----
MIIBCgKCAQEAldbmm4HzEx0elQPnvdhJ3JCrz5AMigVTK+rSqNbB7X9cs52lQ++7
qRja0C1R/uxo5FJrt1yHKh8HOqzgP1HEvcq8rQKNjc08IlXlRc/HXKXWDXE/edn3
4LP0oDWZ2Wf4bDSguuqjyRrbDgYKmiCx3QUVdj29SykIPCfm7L7KRuMZE/qWV79e
0lO10zk+ER72PQg0ddB63kFgtT2FmvCRXxQbI10UGmLfBEF9qvEI1V0aS8Tg04Vb
Bza9YCptmJbl3X44sO0BeHmgzYhm3zGERns83Fqt7RXNrh+pTdKFwyOt55v403wf
QitEZHRYxHSh3VCuw5kobAXlqhcjSdxq4wIDAQAB
-----END RSA PUBLIC KEY-----
-----BEGIN RSA PUBLIC KEY-----
MIIBCgKCAQEAlSXR7MiSp8HpY3BekKnV1Ywsfzx99x1cNjgWetnsIQjL6m8GGxh3
i+zAY341cVQXiNSITSSccUUhWUvcrFLX9LfhqM1/xYf8s69Op9RKydrtNjNfMznN
xLebt3jSQ9E3vsvezOaZube9vgqZYipAHE1cQjIhfBQKHBEHYmvurvuVjj8hWVcz
FaHhpCAiJ1BtT84pwYGABg0RLF9Xl7myabbw9CiRVXj0lN2yujeQ1tSII4Lc6m6W
etctyJ7ysAkNLX1saQ1KA1vomwU8KIBN0tbrX5HTlEav35VijBjjBIqTYiwappnL
TOilQE8X+XDhI3WFkyvt1Vrd5hAVGltN3QIDAQAB
-----END RSA PUBLIC KEY-----
.EP....t..a-..w.re?1\.t...m0%{j..Q.MF.e.....%......=.8..I.n..B.9....!?K.4...    sv-..'jO.9;-.YA.._0...U.
2.)...m^*...c.WX..Nk.S{z1.2...Z..N.g....'....L...U1aA._...*^...[[U..t...=..Dc.S..8..)I..2S....f2....Q4L......4Kf..s.E.c.du.)R.MX.
8.Y........:%.iE..Um..G2.D
```

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