RSA ASSIGNMENT

Name: Ahmed Sayed Sayed

SEC: 1

BN: 3

CODE: 9202111

Chating:

In the chat I used one server- multi client technique with multi threading server and each client is also multi threading .

SERVER:

the server work with multi-threading where the main thred whill start the socket connection with the clients , and when any client ask to connect , the main thread will accept and create another thread to deal with this client

```
9   HEADER = 64
10  # step 1 define the port we will work on
11  PORT = 5050
12  # STEP 2 get the name of the server
13  SERVER = socket.gethostbyname(socket.gethostname())
14  ADDR = (SERVER, PORT)
15  # bind the socket
16  server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
17  # start the communication and wait for clints
18  server.bind(ADDR)
19  FORMAT = "utf-8"
20  DISCONECTE_MESSAGE = "!DISCONNECT"
```

```
def start():
    server.listen()
    print("[LISTENING] server is listening on ", SERVER, flush=True)
    while True:
        coon, addr = server.accept()
        print(coon, flush=True)
        thread = threading.Thread(target=handle_client, args=(coon, addr))
        thread.start()
        print(f"[ACTIVE CONNECTIONS] {len(clients)}", flush=True)


print("STARTING server is starting.....")
start()
```

Then , each client has a single thread that represent him at the server . This thread start by taking the Publick Key (e,N) from the client and broad cast it to all others client (I work with assumption that the server will serve only two clients so I didn't handle the corner cases that will happen when the #clients increased), afther that the thread will wait untill the client send any thing to it , and it will broad cast it to all other clients listen to the server.

By this way the scenario will be as following:

Client 1 connect to the server and send its public key to the server

Client 2 connect to the server and send its public key to the server

Server send the public key of client 1 to client 2

Server send the public key of client 2 to client 1

When client 1 send message to the server, the server receive and send to client 2

When client 2 send message to the server, the server receive and send to client 1

```python
def handle_client(conn, addr):
    print("[NEW CONNECTION] ", addr, " connected.", flush=True)
    connected = True
    add_client(conn)
    spread_public_keys()
    print('finish spread keys', flush=True)
    while connected:
        msg_length = conn.recv(HEADER).decode(FORMAT)
        if msg_length:
            msg_length = int(msg_length)
            msg = conn.recv(msg_length).decode(FORMAT)
            if msg == DISCONECTE_MESSAGE:
                connected = False
                remove_client(conn)
                print("remove client ", addr, flush=True)
                print(f"[ACTIVE CONNECTIONS] {len(clients)}", flush=True)
            else:
                print(f"[{addr}] {msg}", flush=True)
                send_global(str(msg), conn)
            # conn.send("Msg recived".encode(FORMAT))
```

```python
def spread_public_keys():
    for i in range(len(clients)):
        client = clients[i]
        for j in range(len(clients)):
            if(i != j):
                send(client, str(clients_e[j]))
                send(client, str(clients_N[j]))


def add_client(client):

    e = int(recive(client))
    N = int(recive(client))
    clients.append(client)
    clients_e.append(e)
    clients_N.append(N)
    print('client ', client, 'has public key e= '+str(e), ', N= '+str(N))
    print(f"total number er of clints now is : {len(clients)}")
```

```python
def recive(client):
    msg_length = int(client.recv(HEADER).decode(FORMAT))
    e = int(client.recv(msg_length).decode(FORMAT))
    return e


def send(client, msg):
    message = msg.encode(FORMAT)
    msg_length = len(message)
    send_length = str(msg_length).encode(FORMAT)
    send_length += b' ' * (HEADER - len(send_length))
    client.send(send_length)
    client.send(message)


def send_global(message, sender):
    for client in clients:
        if(client != sender):
            print(message, type(message))
            send(client, message)
            # client.send(message.encode(FORMAT))
```

Client:

The client is the chat it-selve, it could be run more than one instance and chat between them.

The client is two threaded program, the first thread is the main thread .

It stablish the connection with the server and create the public key(e,N) and the private key (d,N) , and get the public key of the second Chater and after that start the second thread. And the first thread enter busy loop to read from user and

process the message somehow (will explain later), after that encrypt the message and then send it to the server.

```
HEADER = 64
PORT = 5050
FORMAT = "utf-8"
DISCONECTE_MESSAGE = "!DISCONNECT"
SERVER = socket.gethostbyname(socket.gethostname())

ADDR = (SERVER, PORT)
# bind the socket
ciennt = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# connect the client to the server
ciennt.connect(ADDR)

finish_get_keys = False
```

```
# init the bublic and private key of the
p, q, N, phai, e, d = init_RSA()
print("the parameters of the RSA is:", flush=True)
print(f"p={p}, q={q}, N={N}, phai={phai}, e={e}, d={d}", flush=True)

recive_thread = threading.Thread(target=handle_recive, args=(e, d, N, ciennt))
recive_thread.daemon = True
recive_thread.start()

# befor sending i need to get the public key of the other side
print('wait for server to give the public keys', flush=True)
reciver_e, reciver_N = get_public_key()
finish_get_keys = True
print(
    f"the public key (e,N) of the other side=({reciver_e},{reciver_N}) ", flush=True)

while True:
    message = input()
    if(message == "~~"):
        break
    message = modify_message(message)
    numeric_sequnce = get_numeric_sequnce(message)
    send(str(len(numeric_sequnce)))
    for i in range(len(numeric_sequnce)):
        encripted_num = encript(numeric_sequnce[i], reciver_e, reciver_N)
        send(str(encripted_num))
send(DISCONECTE_MESSAGE)
```

The second thread send the public key to the server , enter busy loop to wait server to send anything , then decrypt it and, extract the message from the decrypted number(will explain latter)and print to the user

```python
def handle_recive(e, d, N, ciennt):
    # befor recive i need to send my public key to the other side
    send_public_key(e, N)
    while (finish_get_keys == False):
        ds = 12
    print('start reciving', flush=True)
    while True:
        sequence_len = int(recive())
        print(sequence_len)
        message = ''
        for i in range(sequence_len):
            encripted_num = int(recive())
            dec_message = decript(encripted_num, d, N)
            message = message+extract_message(dec_message)
        print("recived:"+message, flush=True)
```

How to send the message?

1:- read from user the whole message

2:- modify the message by make its length divisible by 5 and replace any thing mot in (small alphabetic – numbers (0~9) - space) to space

```python
def modify_message(message):
    length = len(message)
    new_string = ' '
    for i in range(length):
        find1 = message[i].isalpha()
        find2 = message[i].isnumeric()
        find3 = message[i] == ' '
        if (not (find1 or find2 or find3)):
            message = message.replace(message[i], new_string)
    if (len(message) % 5 != 0):
        reminder = 5-len(message) % 5
        message = message+' '*reminder
    return message
```

3:- divide the message into blocks , each one 5 characters ,then get number represent each block , and send the number of blocks to the server

```python
def get_numeric_sequnce(message):
    sequence = []
    sequence_size = len(message)//5
    for i in range(sequence_size):
        block = message[5*i:5*i+5]
        block_number = 0
        for j in range(5):
            block_number += pow(37, j) * alphabet_conversion(block[j])
        sequence.append(block_number)
    return sequence
```

```python
def alphabet_conversion(char):
    out = 0
    find1 = char.isnumeric()
    find2 = char.isalpha()
    find3 = char == ' '
    if (find1):
        out = int(char)
    elif (find2):
        out = ord(char)-ord('a')+10
    elif (find3):
        out = 36
    return out
```

4:- encrypt each block with the public key of the second side and send it to the server

```python
def encript(num, reciver_e, reciver_N):
    encripted_num = pow(num, reciver_e, reciver_N)
    return encripted_num
```

How to receive message?

1:- wait until server send any thing , and I know that the first recived message is the number of blocks in the message

2:- loop in the number of blocks and recive each block from the server

3:- decript the block with my private key

```python
def decript(encripted_num, d, N):
    dec_message = pow(encripted_num, d, N)
    return dec_message
```

4:- extract the block message from the decrypted number

```python
def extract_message(dec_message):
    c0, c1, c2, c3, c4 = '', '', '', '', ''

    num = dec_message % 37
    c0 = alpahbet_extraction(num)

    dec_message = (dec_message-num)//37
    num = dec_message % 37
    c1 = alpahbet_extraction(num)

    dec_message = (dec_message-num)//37
    num = dec_message % 37
    c2 = alpahbet_extraction(num)

    dec_message = (dec_message-num)//37
    num = dec_message % 37
    c3 = alpahbet_extraction(num)

    dec_message = (dec_message-num)//37
    num = dec_message % 37
    c4 = alpahbet_extraction(num)

    message = c0+c1+c2+c3+c4
    return message
```

```
def alpahbet_extraction(num):
    c = ''
    if(num < 10):
        c = str(num)
    elif(num < 36):
        num -= 10
        num += ord('a')
        c = chr(num)
    else:
        c = ' '
    return c
```

5:- add all the blocks into one message and print it to the user

Attack to break the RSA algorithm

Lets assume we know the public key of the user (e,N)

How to get its private key (p,q,phai,d)?

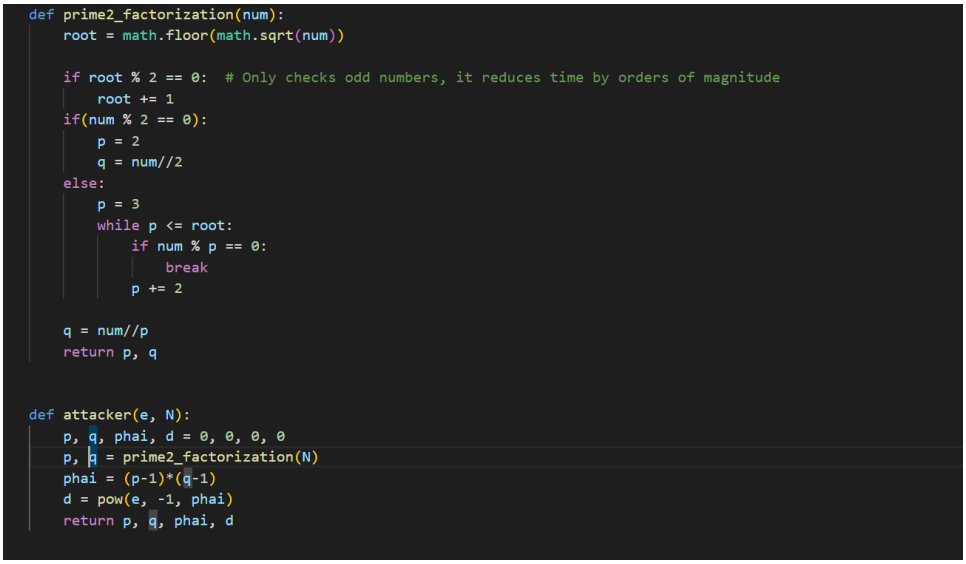
1-: ge could get the primes p,q from prime factorization of N

```
def prime2_factorization(num):
    root = math.floor(math.sqrt(num))

    if root % 2 == 0:  # Only checks odd numbers, it reduces time by orders of magnitude
        root += 1
    if(num % 2 == 0):
        p = 2
        q = num//2
    else:
        p = 3
        while p <= root:
            if num % p == 0:
                break
            p += 2

    q = num//p
    return p, q
```

2-: phai(N) = (p-1)*(q-1)

3:- d=e^-1mod phai

The whole code :



```python
def prime2_factorization(num):
    root = math.floor(math.sqrt(num))

    if root % 2 == 0:  # Only checks odd numbers, it reduces time by orders of magnitude
        root += 1
    if(num % 2 == 0):
        p = 2
        q = num//2
    else:
        p = 3
        while p <= root:
            if num % p == 0:
                break
            p += 2

    q = num//p
    return p, q


def attacker(e, N):
    p, q, phai, d = 0, 0, 0, 0
    p, q = prime2_factorization(N)
    phai = (p-1)*(q-1)
    d = pow(e, -1, phai)
    return p, q, phai, d
```

RSA Implement :

1-: take the size of the public key (N) from the user

2:- calc the size of p,q and choose its value randomly

3:- N=p*q

4:-phai(N)= (p-1)*(q-1)

5:- select e randomly such that gcd(e,phai)=1

6:- d=e^-1 mod phai(N)

```python
def init_RSA():
    bit_size = int(
        input("Please enter the size of the publick key N in bits: "))
    bit_size_p = bit_size//2
    bit_size_q = bit_size-bit_size_p

    lower_range_p = 1 << (bit_size_p-1)
    upper_range_p = sum(1 << i for i in range(bit_size_p))
    p = sympy.randprime(lower_range_p, upper_range_p)

    lower_range_q = 1 << (bit_size_q-1)
    upper_range_q = sum(1 << i for i in range(bit_size_q))
    q = sympy.randprime(lower_range_q, upper_range_q)
    N = p*q
    phai = (p-1)*(q-1)
    e = random.randint(2, phai)
    temp = math.gcd(phai, e)
    while temp > 1:
        e = e//temp
        temp = math.gcd(phai, e)
    d = pow(e, -1, phai)
    return p, q, N, phai, e, d
```

Time analysis of the attaker

I used brute force technique to find the prime factorization of the public key N knowing that N consist of only two primes, so when the size increased by 1 , the vlaue increased by multiple of 2 -> then the complxity of the prime factorization is O(2^(size of prime N)) Wich is exponential