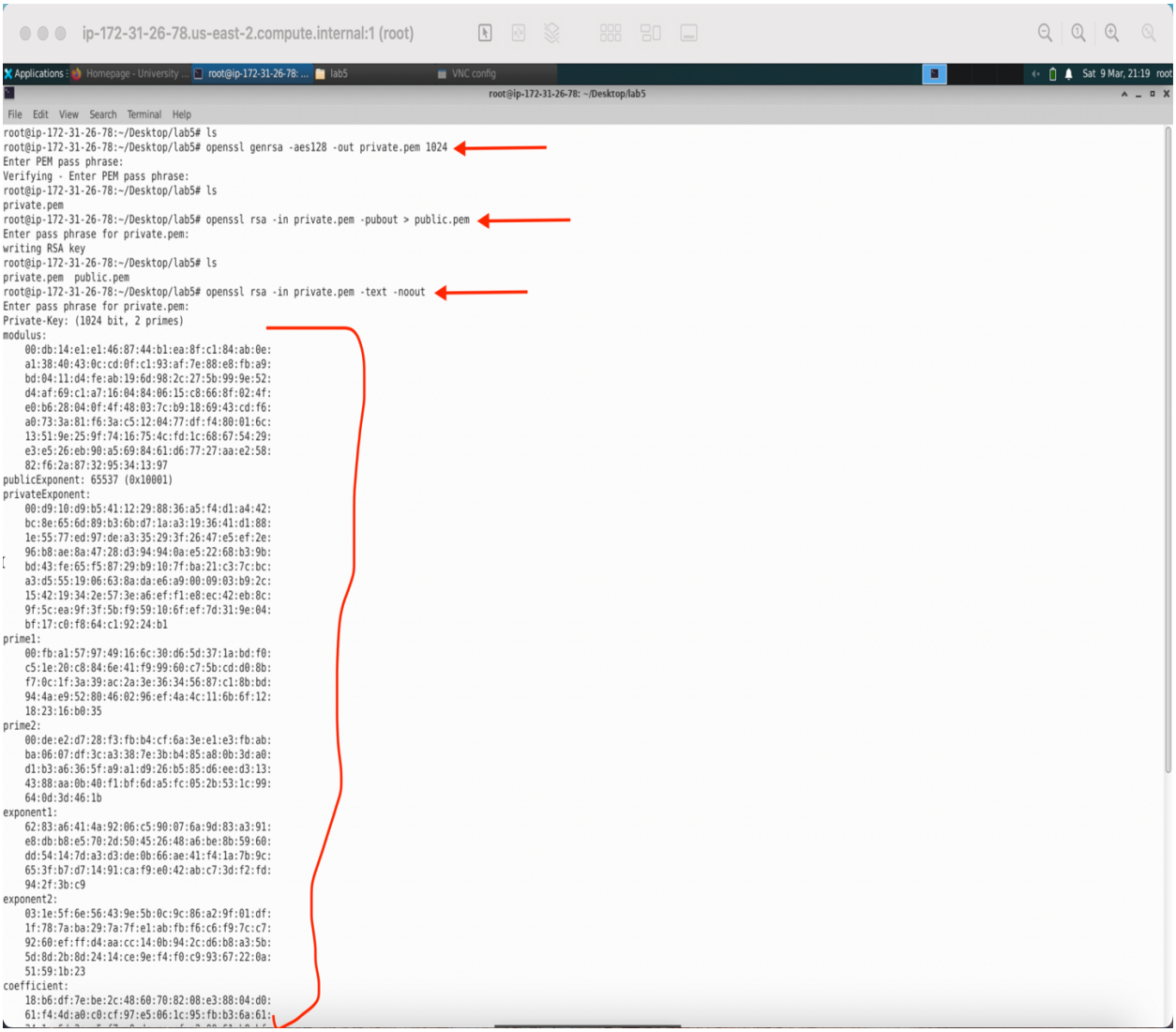
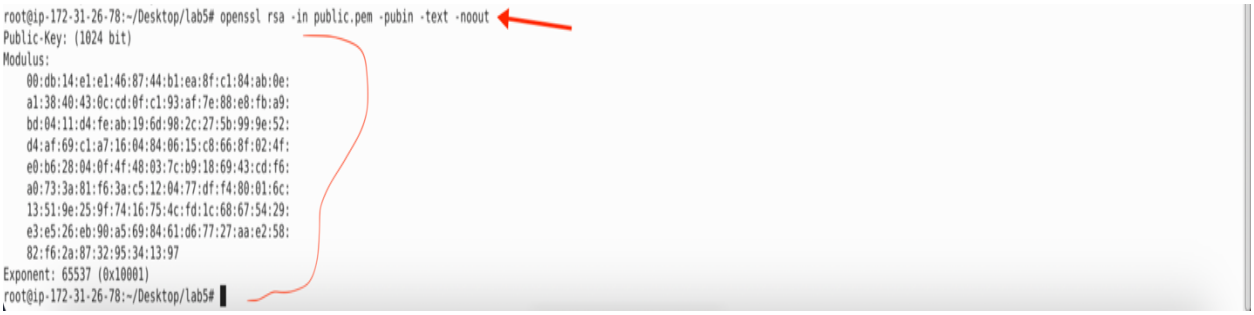
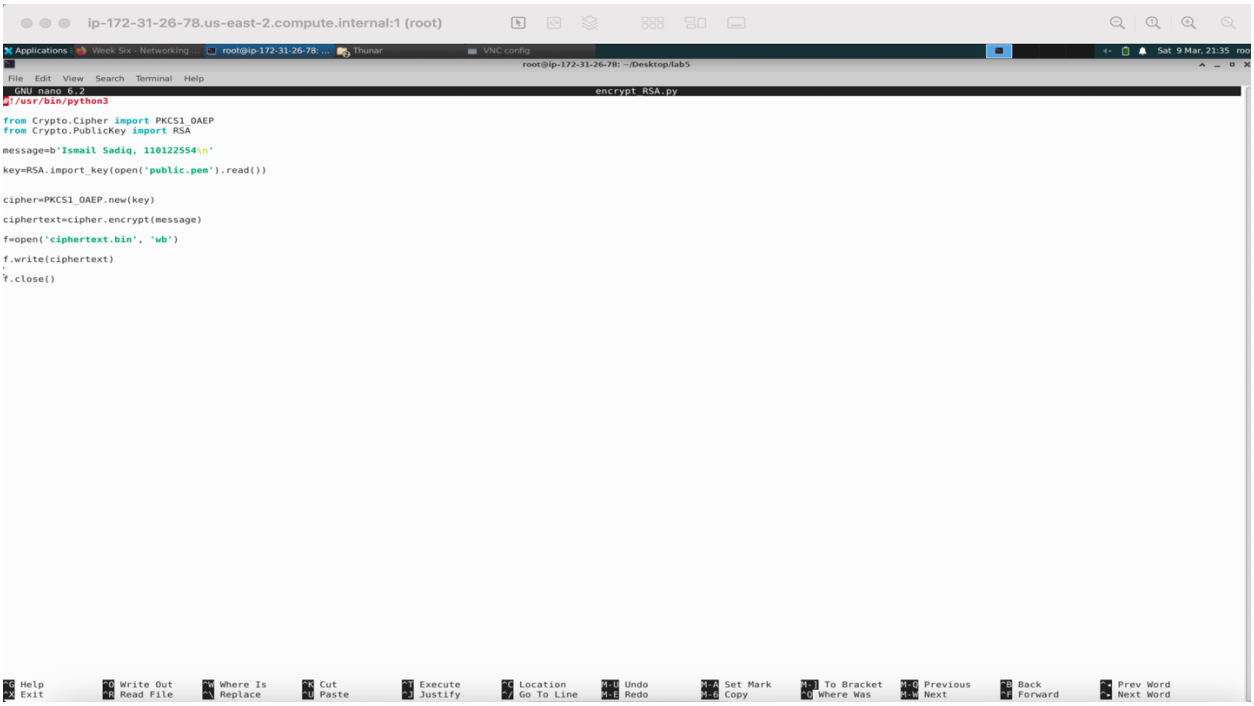
**ISMAIL SADIQ** **110122554** **LAB5**

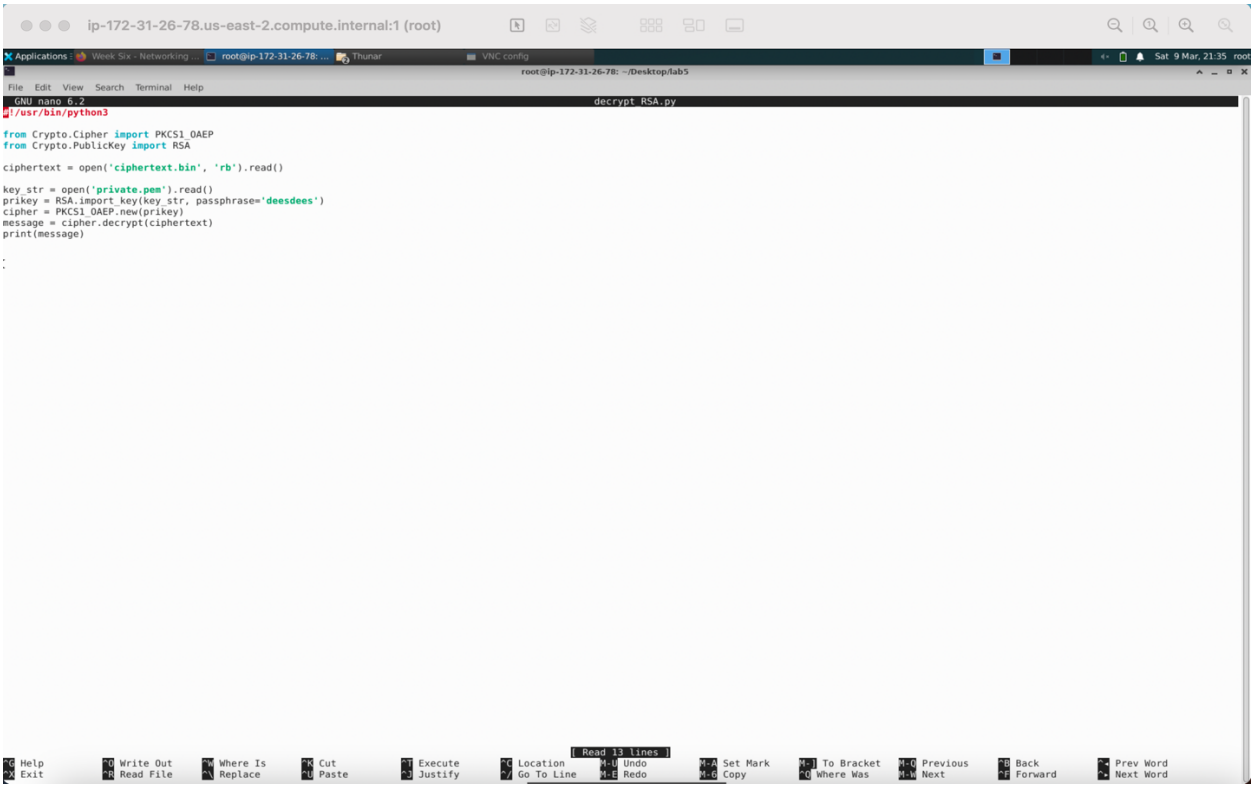
1. Use openssl to generate RSA public/private key:

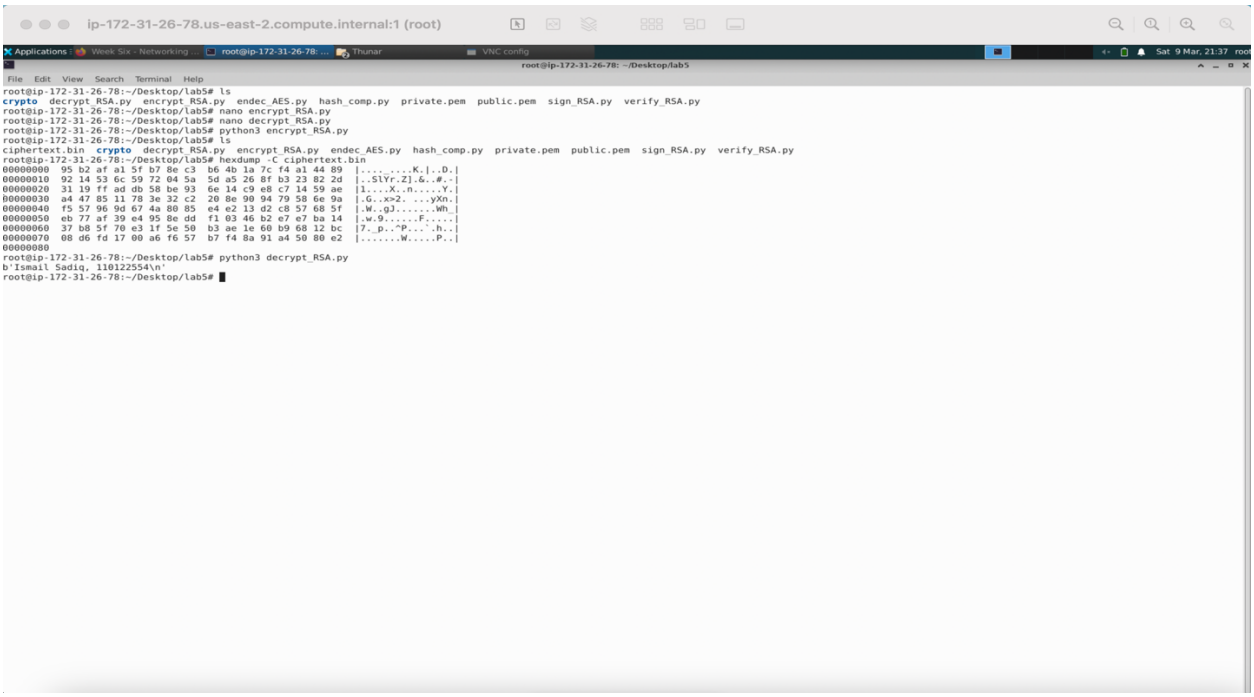


**2.** In this problem, you need to practice RSA encryption and decryption:






**3.** (optional)

**4.** In this problem, you will use Diffie-Hellman with authentication to protect the client-server communication. Implement the following functionalities:

**Server.py**:

```
import socket
from time import sleep
from Crypto.Random.random import getrandbits
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES
from Crypto.Hash import SHA256, MD5
from Crypto.Util.Padding import pad, unpad

def diffie_hellman():
    p = 25822498780869085896559191720030118743297057928292235128306593565406476220168411946296453532801378314
35903171972747559779
    g = 2
    x = getrandbits(400)
    X = pow(g, x, p)
    return p, g, x, X

def key_exchange(sock):
    p, g, x, X = diffie_hellman()
    sock.send(p.to_bytes(256, 'big'))
    sock.send(g.to_bytes(256, 'big'))
    sock.send(X.to_bytes(256, 'big'))
    Y = int.from_bytes(sock.recv(256), 'big')
    shared_key = pow(Y, x, p)
    shared_key = shared_key.to_bytes(128, 'big')
    shared_key = MD5.new(shared_key).digest()
    return shared_key

"server.py" 79L, 2450C                                                    1,1           Top
```

```python
def aes_encrypt(message, key):
    iv = get_random_bytes(16)
    cipher = AES.new(key, AES.MODE_CBC, iv=iv)
    ciphertext = cipher.encrypt(pad(message.encode('utf-8'), 16))
    tag = SHA256.new(ciphertext).digest()
    return ciphertext, tag, iv

def aes_decrypt(ciphertext, key, tag, iv):
    cipher = AES.new(key, AES.MODE_CBC, iv=iv)
    decrypted = unpad(cipher.decrypt(ciphertext), 16)
    if SHA256.new(ciphertext).digest() != tag:
        raise Exception("Message integrity verification failed!")
    return decrypted.decode('utf-8')

def start_server():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(('127.0.0.1', 12345))
    server.listen(1)

    print("Server listening on port 12345...")

    connection, address = server.accept()

    print(f"Connection from {address}")

    shared_key = key_exchange(connection)
```

47,0-1 50%

```python
    while True:
        message = input("You: ")
        if message.lower() == 'exit':
            break

        ciphertext, tag, iv = aes_encrypt(message, shared_key)
        connection.send(ciphertext)
        sleep(1)
        connection.send(tag)
        sleep(1)
        connection.send(iv)

        received_ciphertext = connection.recv(1024)
        received_tag = connection.recv(32)
        received_iv = connection.recv(32)
        try:
            received_message = aes_decrypt(received_ciphertext, shared_key, received_tag, received_iv)
            print(f"Client: {received_message}")
        except Exception as e:
            print(f"Error: {e}")

    connection.close()
    server.close()

if __name__ == "__main__":
    start_server()
```

77,0-1 Bot

**Client.py**:

```
import socket
from time import sleep
from Crypto.Random.random import getrandbits
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES
from Crypto.Hash import SHA256, MD5
from Crypto.Util.Padding import pad, unpad

def diffie_hellman(sock):
    p = int.from_bytes(sock.recv(256), 'big')
    g = int.from_bytes(sock.recv(256), 'big')
    Y = int.from_bytes(sock.recv(256), 'big')
    x = getrandbits(400)
    X = pow(g, x, p)
    sock.send(X.to_bytes(256, 'big'))
    shared_key = pow(Y, x, p)
    shared_key = shared_key.to_bytes(128, 'big')
    shared_key = MD5.new(shared_key).digest()
    return shared_key

def aes_encrypt(message, key):
    iv = get_random_bytes(16)
    cipher = AES.new(key, AES.MODE_CBC, iv=iv)
    ciphertext = cipher.encrypt(pad(message.encode('utf-8'), 16))
    tag = SHA256.new(ciphertext).digest()
    return ciphertext, tag, iv
```

1,1 Top

```
def aes_decrypt(ciphertext, key, tag, iv):
    cipher = AES.new(key, AES.MODE_CBC, iv=iv)
    decrypted = unpad(cipher.decrypt(ciphertext), 16)
    if SHA256.new(ciphertext).digest() != tag:
        raise Exception("Message integrity verification failed!")

    return decrypted.decode('utf-8')

def start_client():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(('127.0.0.1', 12345))
    shared_key = diffie_hellman(client)
    while True:
        received_ciphertext = client.recv(1024)
        received_tag = client.recv(32)
        received_iv = client.recv(32)

        received_message = aes_decrypt(received_ciphertext, shared_key, received_tag, received_iv)
        print(f"Ciphertext: {received_ciphertext}\nTag: {received_tag}\nShared key: {shared_key}\nDecrypted t
ext: {received_message}")

        print(f"Server: {received_message}")

        message = input("You: ")

        if message.lower() == 'exit':
            break
```

47,0-1 69%

```
    while True:
        received_ciphertext = client.recv(1024)
        received_tag = client.recv(32)
        received_iv = client.recv(32)

        received_message = aes_decrypt(received_ciphertext, shared_key, received_tag, received_iv)
        print(f"Ciphertext: {received_ciphertext}\nTag: {received_tag}\nShared key: {shared_key}\nDecrypted t
ext: {received_message}")

        print(f"Server: {received_message}")

        message = input("You: ")

        if message.lower() == 'exit':
            break

        ciphertext, tag, iv = aes_encrypt(message, shared_key)
        client.send(ciphertext)
        sleep(1)
        client.send(tag)
        sleep(1)
        client.send(iv)

    client.close()

if __name__ == "__main__":
    start_client()
                                                                                65,1          Bot
```

**Output**:

```
[03/08/24]seed@VM:~/.../lab5-doc$ python3 server.py
Server listening on port 12345...
Connection from ('127.0.0.1', 38964)
You: hi there, how are you
Client: oh wow, long time, I'm good
You: this is an encrpyted text
```

```
[03/08/24]seed@VM:~/.../lab5-doc$ python3 client.py
Ciphertext: b'\xc0Z\xd8\xa1\xc5\x12h.\xe8B\x94\x8eY+\x9a(\x86|\xb2b\x95\x18\xbc\x8cN\xedl\x19I\xc7!('
Tag: b'\x0cK\xcdm[\xd5\x80.\xd04e\x8f\xbf?j\xdfL\xb3,\x8e\x08 a\x04\x1e\xf3\xfd@x\xc9\xb1\n'
Shared key: b'rW\xa9z\x0cH\xf5p\x85Nh\x1bx\xfd\xb7y'
Decrypted text: hi there, how are you
Server: hi there, how are you
You: oh wow, long time, I'm good
Ciphertext: b'\x9b`xrS\x1aTq\x89\x8b\x06\x83:;Qd\xf4y\xbb\x96\xd3\xd1Q\x06\n"\x82za<"]'
Tag: b'\t\xe1J2wZ\xd9\xf5l\x88\xad\x08\x91%\xbao\r\xe2\x9f|\xf0\x87\x8f\xc9^\xe6\xcc\xdd{\x0fR\xc6'
Shared key: b'rW\xa9z\x0cH\xf5p\x85Nh\x1bx\xfd\xb7y'
Decrypted text: this is an encrpyted text
Server: this is an encrpyted text
You:
```