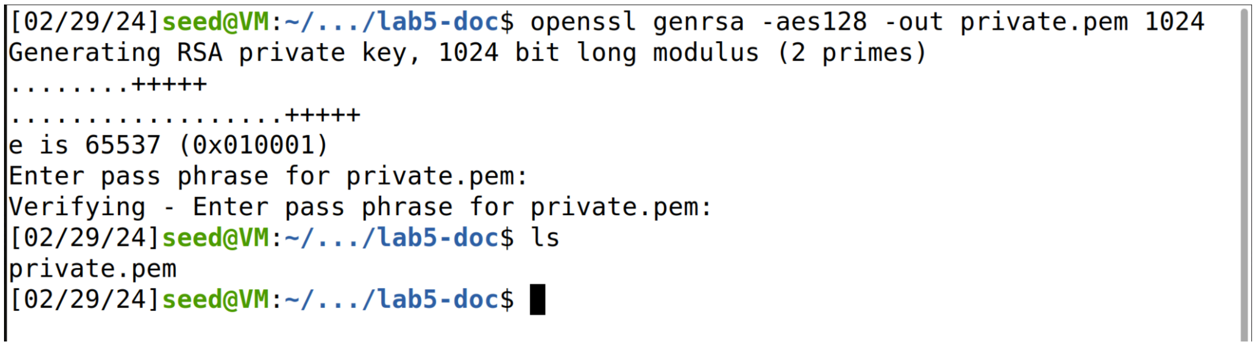
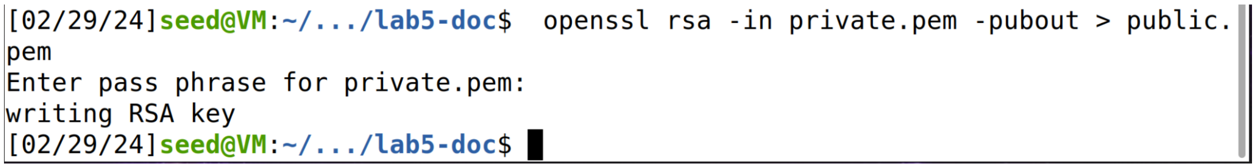
Lab 5
COMP8677 - Networking and Data Security
Joel Mathew Koshy - 110123206

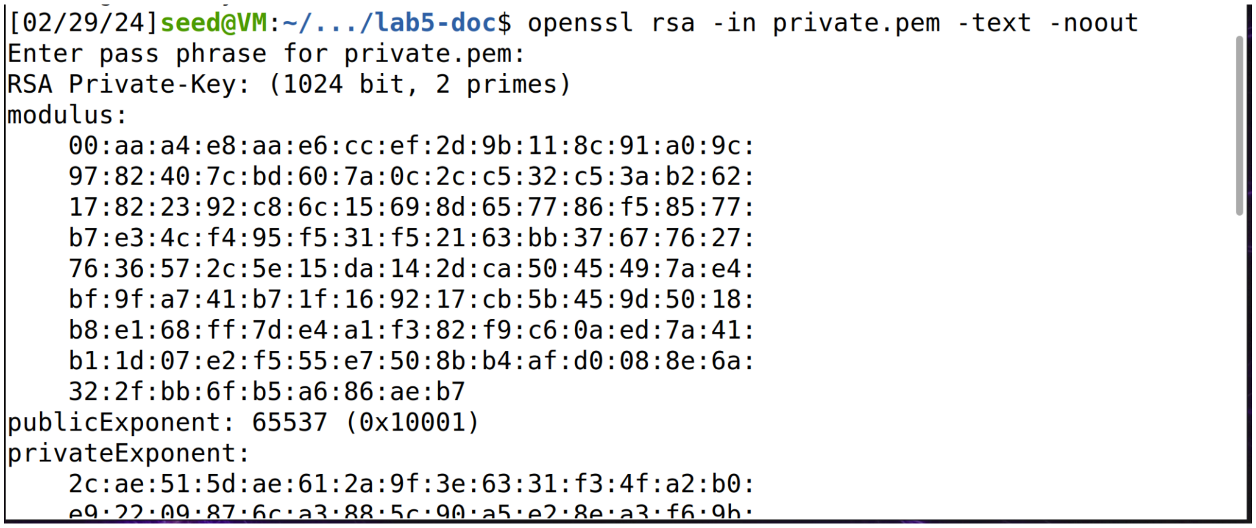
1. Use openssl to generate RSA public/private key



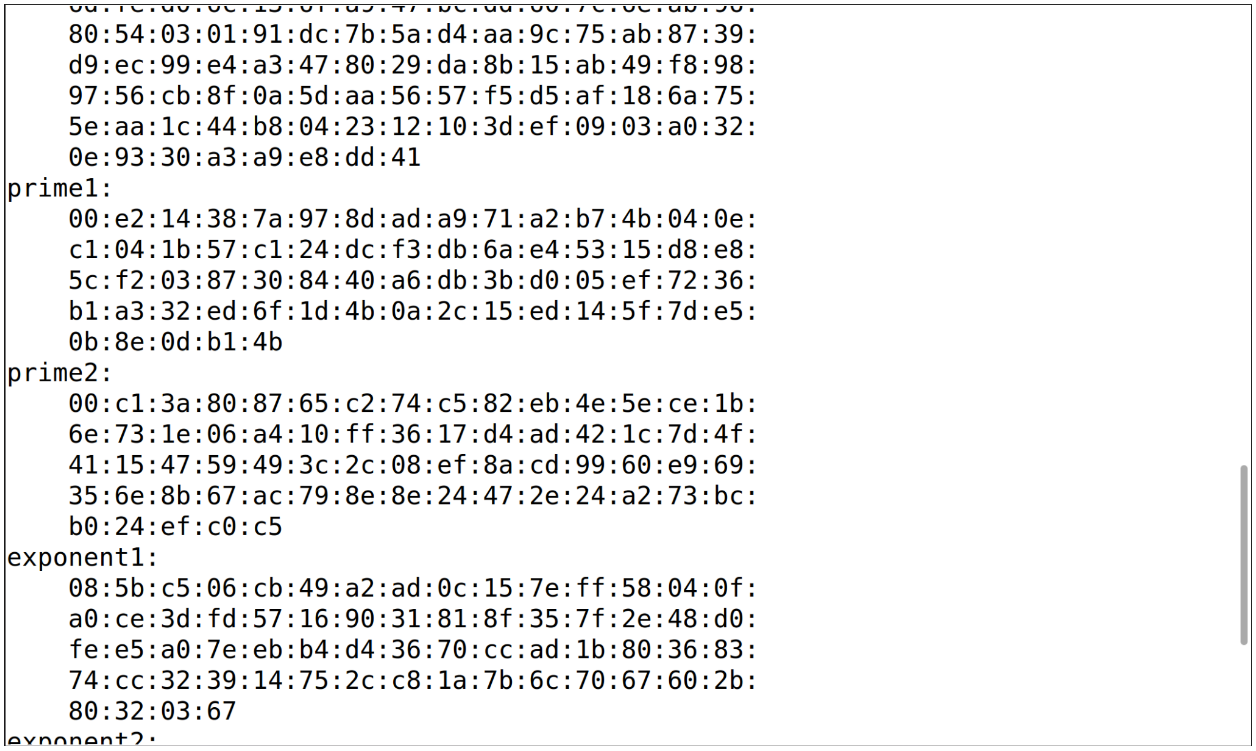
```
[02/29/24]seed@VM:~/.../lab5-doc$ openssl genrsa -aes128 -out private.pem 1024
Generating RSA private key, 1024 bit long modulus (2 primes)
........+++++
..................+++++
e is 65537 (0x010001)
Enter pass phrase for private.pem:
Verifying - Enter pass phrase for private.pem:
[02/29/24]seed@VM:~/.../lab5-doc$ ls
private.pem
[02/29/24]seed@VM:~/.../lab5-doc$ 
```



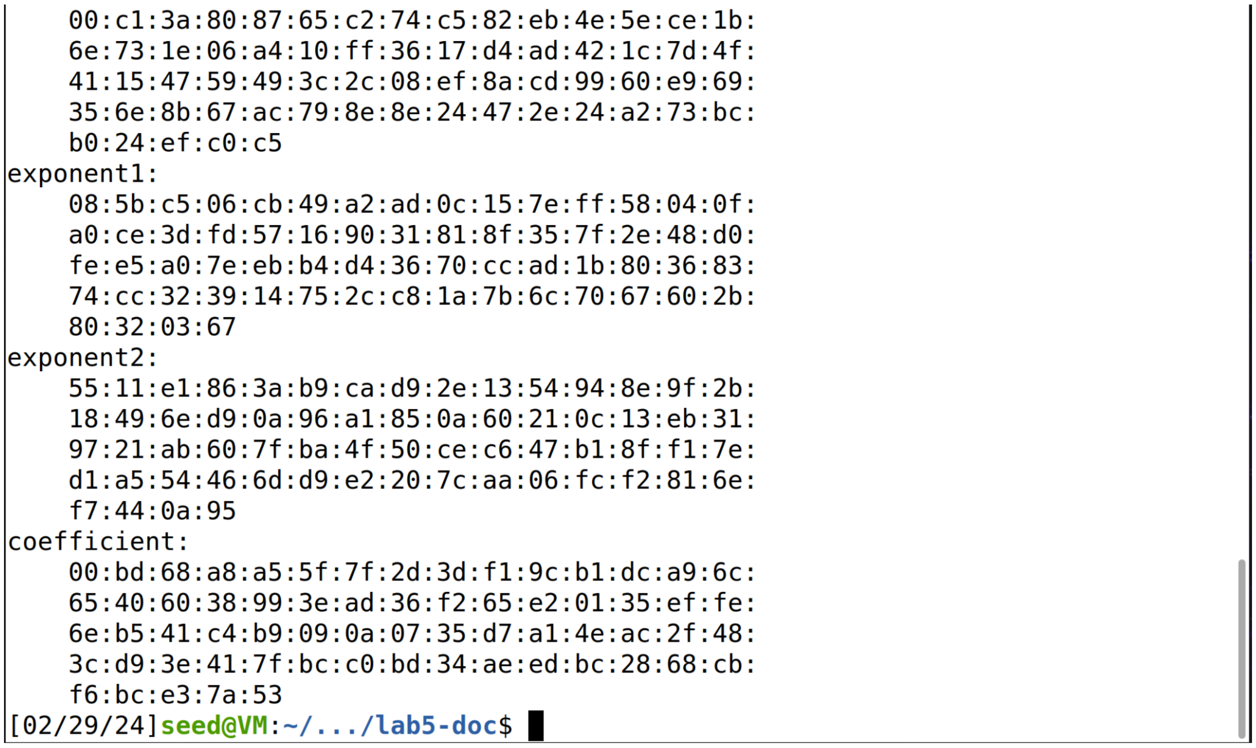
```
[02/29/24]seed@VM:~/.../lab5-doc$  openssl rsa -in private.pem -pubout > public.
pem
Enter pass phrase for private.pem:
writing RSA key
[02/29/24]seed@VM:~/.../lab5-doc$ 
```



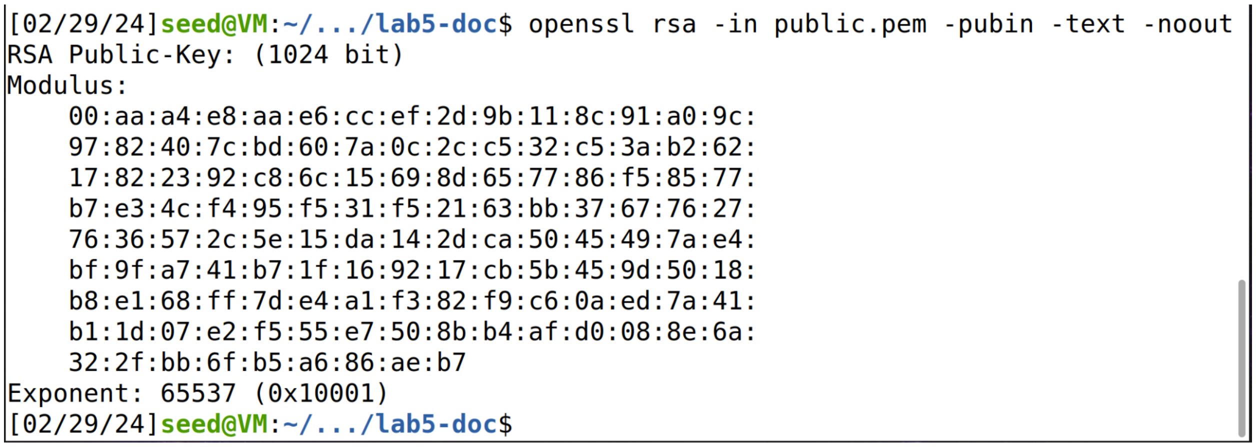
```
[02/29/24]seed@VM:~/.../lab5-doc$ openssl rsa -in private.pem -text -noout
Enter pass phrase for private.pem:
RSA Private-Key: (1024 bit, 2 primes)
modulus:
    00:aa:a4:e8:aa:e6:cc:ef:2d:9b:11:8c:91:a0:9c:
    97:82:40:7c:bd:60:7a:0c:2c:c5:32:c5:3a:b2:62:
    17:82:23:92:c8:6c:15:69:8d:65:77:86:f5:85:77:
    b7:e3:4c:f4:95:f5:31:f5:21:63:bb:37:67:76:27:
    76:36:57:2c:5e:15:da:14:2d:ca:50:45:49:7a:e4:
    bf:9f:a7:41:b7:1f:16:92:17:cb:5b:45:9d:50:18:
    b8:e1:68:ff:7d:e4:a1:f3:82:f9:c6:0a:ed:7a:41:
    b1:1d:07:e2:f5:55:e7:50:8b:b4:af:d0:08:8e:6a:
    32:2f:bb:6f:b5:a6:86:ae:b7
publicExponent: 65537 (0x10001)
privateExponent:
    2c:ae:51:5d:ae:61:2a:9f:3e:63:31:f3:4f:a2:b0:
    e9:22:09:87:6c:a3:88:5c:90:a5:e2:8e:a3:f6:9b:
```



```
    80:54:03:01:91:dc:7b:5a:d4:aa:9c:75:ab:87:39:
    d9:ec:99:e4:a3:47:80:29:da:8b:15:ab:49:f8:98:
    97:56:cb:8f:0a:5d:aa:56:57:f5:d5:af:18:6a:75:
    5e:aa:1c:44:b8:04:23:12:10:3d:ef:09:03:a0:32:
    0e:93:30:a3:a9:e8:dd:41
prime1:
    00:e2:14:38:7a:97:8d:ad:a9:71:a2:b7:4b:04:0e:
    c1:04:1b:57:c1:24:dc:f3:db:6a:e4:53:15:d8:e8:
    5c:f2:03:87:30:84:40:a6:db:3b:d0:05:ef:72:36:
    b1:a3:32:ed:6f:1d:4b:0a:2c:15:ed:14:5f:7d:e5:
    0b:8e:0d:b1:4b
prime2:
    00:c1:3a:80:87:65:c2:74:c5:82:eb:4e:5e:ce:1b:
    6e:73:1e:06:a4:10:ff:36:17:d4:ad:42:1c:7d:4f:
    41:15:47:59:49:3c:2c:08:ef:8a:cd:99:60:e9:69:
    35:6e:8b:67:ac:79:8e:8e:24:47:2e:24:a2:73:bc:
    b0:24:ef:c0:c5
exponent1:
    08:5b:c5:06:cb:49:a2:ad:0c:15:7e:ff:58:04:0f:
    a0:ce:3d:fd:57:16:90:31:81:8f:35:7f:2e:48:d0:
    fe:e5:a0:7e:eb:b4:d4:36:70:cc:ad:1b:80:36:83:
    74:cc:32:39:14:75:2c:c8:1a:7b:6c:70:67:60:2b:
    80:32:03:67
```

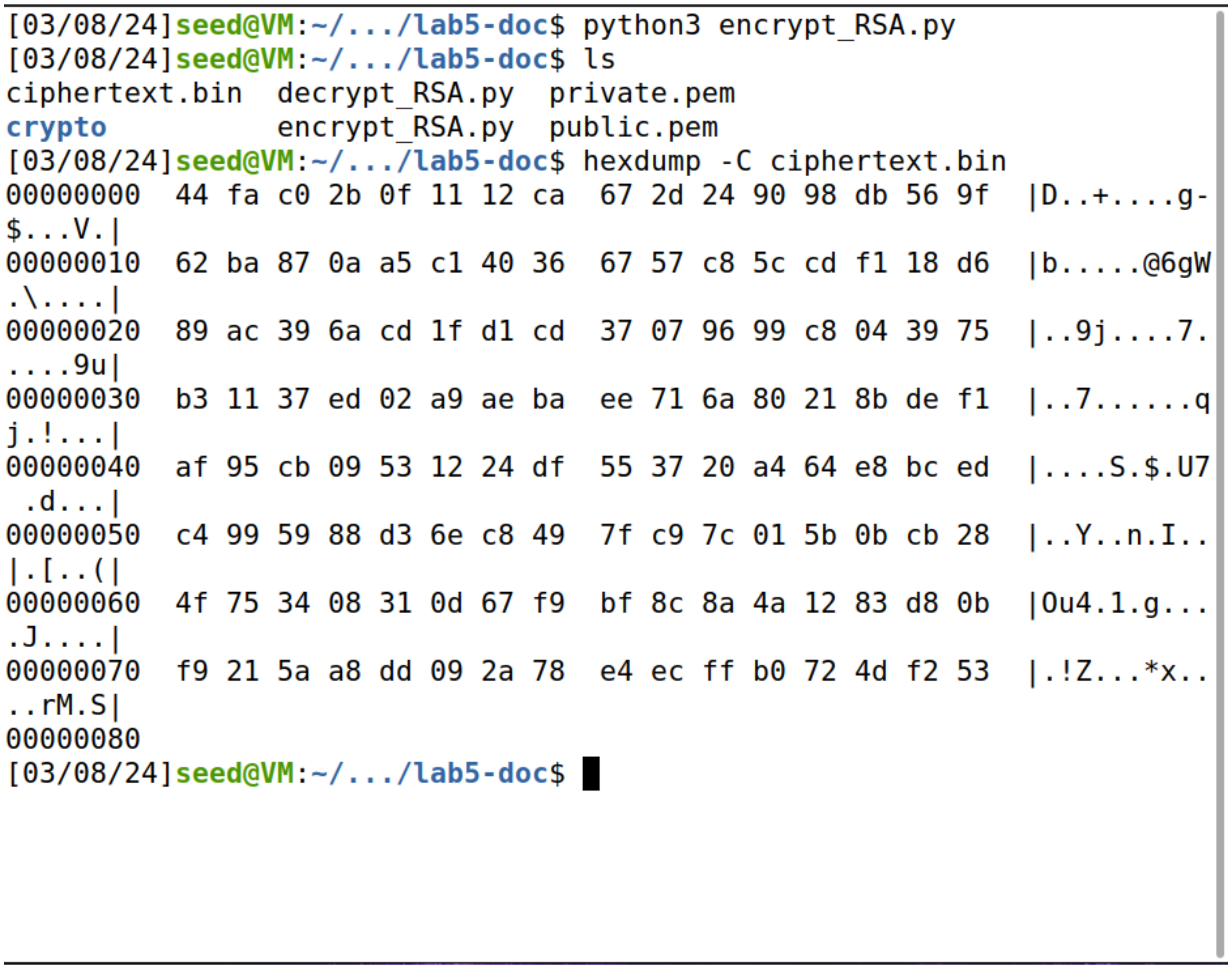


```
    00:c1:3a:80:87:65:c2:74:c5:82:eb:4e:5e:ce:1b:
    6e:73:1e:06:a4:10:ff:36:17:d4:ad:42:1c:7d:4f:
    41:15:47:59:49:3c:2c:08:ef:8a:cd:99:60:e9:69:
    35:6e:8b:67:ac:79:8e:8e:24:47:2e:24:a2:73:bc:
    b0:24:ef:c0:c5
exponent1:
    08:5b:c5:06:cb:49:a2:ad:0c:15:7e:ff:58:04:0f:
    a0:ce:3d:fd:57:16:90:31:81:8f:35:7f:2e:48:d0:
    fe:e5:a0:7e:eb:b4:d4:36:70:cc:ad:1b:80:36:83:
    74:cc:32:39:14:75:2c:c8:1a:7b:6c:70:67:60:2b:
    80:32:03:67
exponent2:
    55:11:e1:86:3a:b9:ca:d9:2e:13:54:94:8e:9f:2b:
    18:49:6e:d9:0a:96:a1:85:0a:60:21:0c:13:eb:31:
    97:21:ab:60:7f:ba:4f:50:ce:c6:47:b1:8f:f1:7e:
    d1:a5:54:46:6d:d9:e2:20:7c:aa:06:fc:f2:81:6e:
    f7:44:0a:95
coefficient:
    00:bd:68:a8:a5:5f:7f:2d:3d:f1:9c:b1:dc:a9:6c:
    65:40:60:38:99:3e:ad:36:f2:65:e2:01:35:ef:fe:
    6e:b5:41:c4:b9:09:0a:07:35:d7:a1:4e:ac:2f:48:
    3c:d9:3e:41:7f:bc:c0:bd:34:ae:ed:bc:28:68:cb:
    f6:bc:e3:7a:53
[02/29/24]seed@VM:~/.../lab5-doc$
```

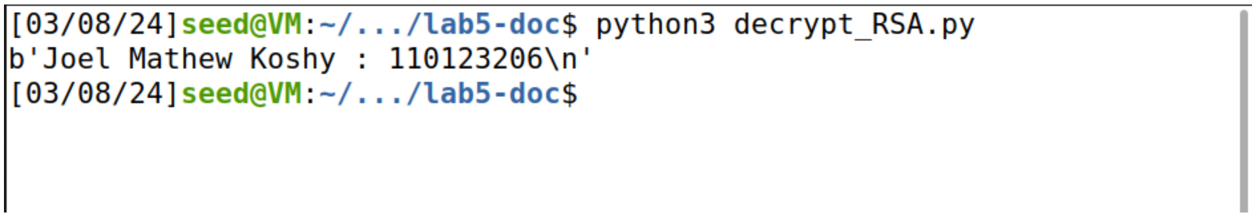


```
[02/29/24]seed@VM:~/.../lab5-doc$ openssl rsa -in public.pem -pubin -text -noout
RSA Public-Key: (1024 bit)
Modulus:
    00:aa:a4:e8:aa:e6:cc:ef:2d:9b:11:8c:91:a0:9c:
    97:82:40:7c:bd:60:7a:0c:2c:c5:32:c5:3a:b2:62:
    17:82:23:92:c8:6c:15:69:8d:65:77:86:f5:85:77:
    b7:e3:4c:f4:95:f5:31:f5:21:63:bb:37:67:76:27:
    76:36:57:2c:5e:15:da:14:2d:ca:50:45:49:7a:e4:
    bf:9f:a7:41:b7:1f:16:92:17:cb:5b:45:9d:50:18:
    b8:e1:68:ff:7d:e4:a1:f3:82:f9:c6:0a:ed:7a:41:
    b1:1d:07:e2:f5:55:e7:50:8b:b4:af:d0:08:8e:6a:
    32:2f:bb:6f:b5:a6:86:ae:b7
Exponent: 65537 (0x10001)
[02/29/24]seed@VM:~/.../lab5-doc$
```

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



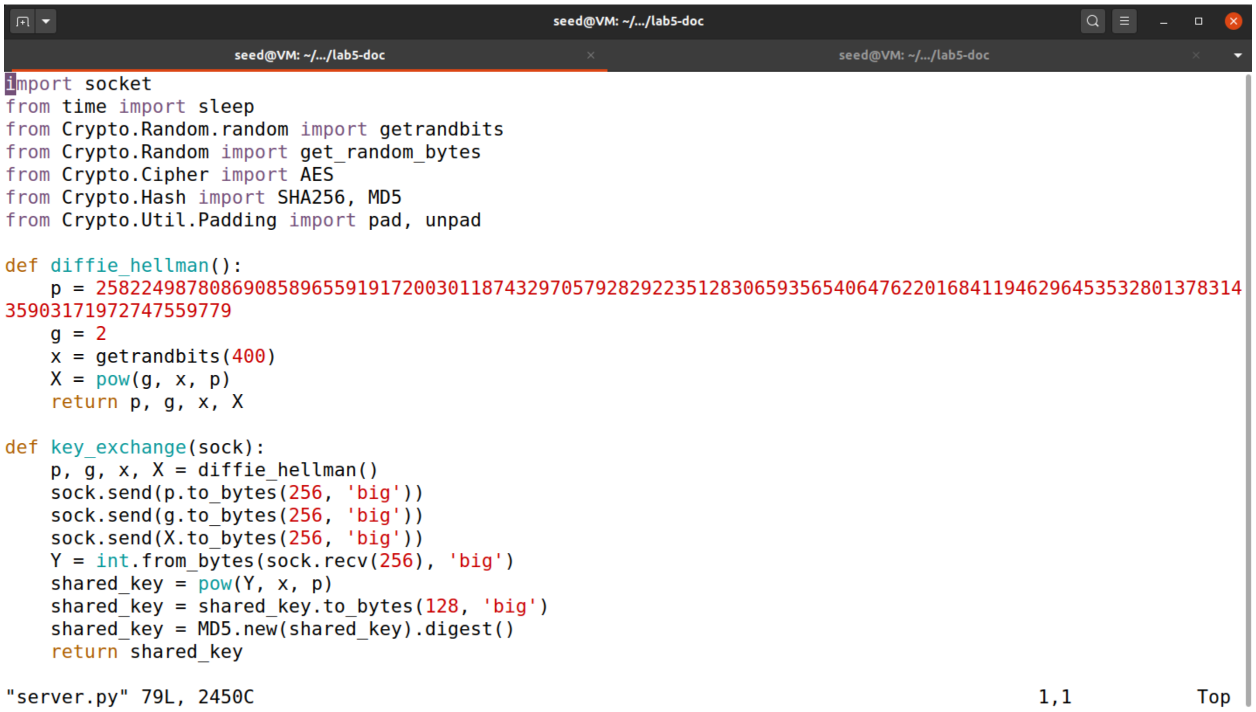
```
[03/08/24]seed@VM:~/.../lab5-doc$ python3 encrypt_RSA.py
[03/08/24]seed@VM:~/.../lab5-doc$ ls
ciphertext.bin  decrypt_RSA.py  private.pem
crypto          encrypt_RSA.py  public.pem
[03/08/24]seed@VM:~/.../lab5-doc$ hexdump -C ciphertext.bin
00000000  44 fa c0 2b 0f 11 12 ca  67 2d 24 90 98 db 56 9f  |D..+....g-
$...V.|
00000010  62 ba 87 0a a5 c1 40 36  67 57 c8 5c cd f1 18 d6  |b.....@6gW
.\....|
00000020  89 ac 39 6a cd 1f d1 cd  37 07 96 99 c8 04 39 75  |..9j....7.
....9u|
00000030  b3 11 37 ed 02 a9 ae ba  ee 71 6a 80 21 8b de f1  |..7......q
j.!...|
00000040  af 95 cb 09 53 12 24 df  55 37 20 a4 64 e8 bc ed  |....S.$.U7
 .d...|
00000050  c4 99 59 88 d3 6e c8 49  7f c9 7c 01 5b 0b cb 28  |..Y..n.I..
|.[..(|
00000060  4f 75 34 08 31 0d 67 f9  bf 8c 8a 4a 12 83 d8 0b  |Ou4.1.g...
.J....|
00000070  f9 21 5a a8 dd 09 2a 78  e4 ec ff b0 72 4d f2 53  |.!Z...*x..
..rM.S|
00000080
[03/08/24]seed@VM:~/.../lab5-doc$
```



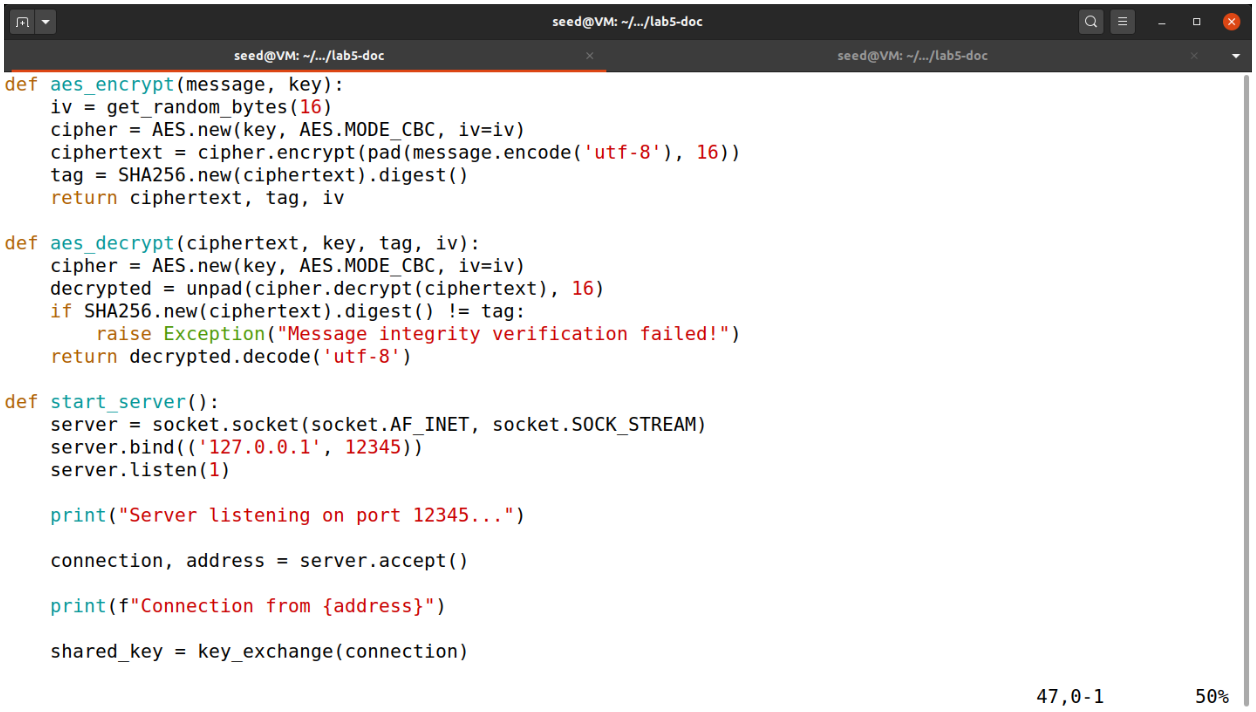
```
[03/08/24]seed@VM:~/.../lab5-doc$ python3 decrypt_RSA.py
b'Joel Mathew Koshy : 110123206\n'
[03/08/24]seed@VM:~/.../lab5-doc$
```

3. **Optional**
4. **In this problem, you will use Diffie-Hellman with authentication to protect the client-server communication.**

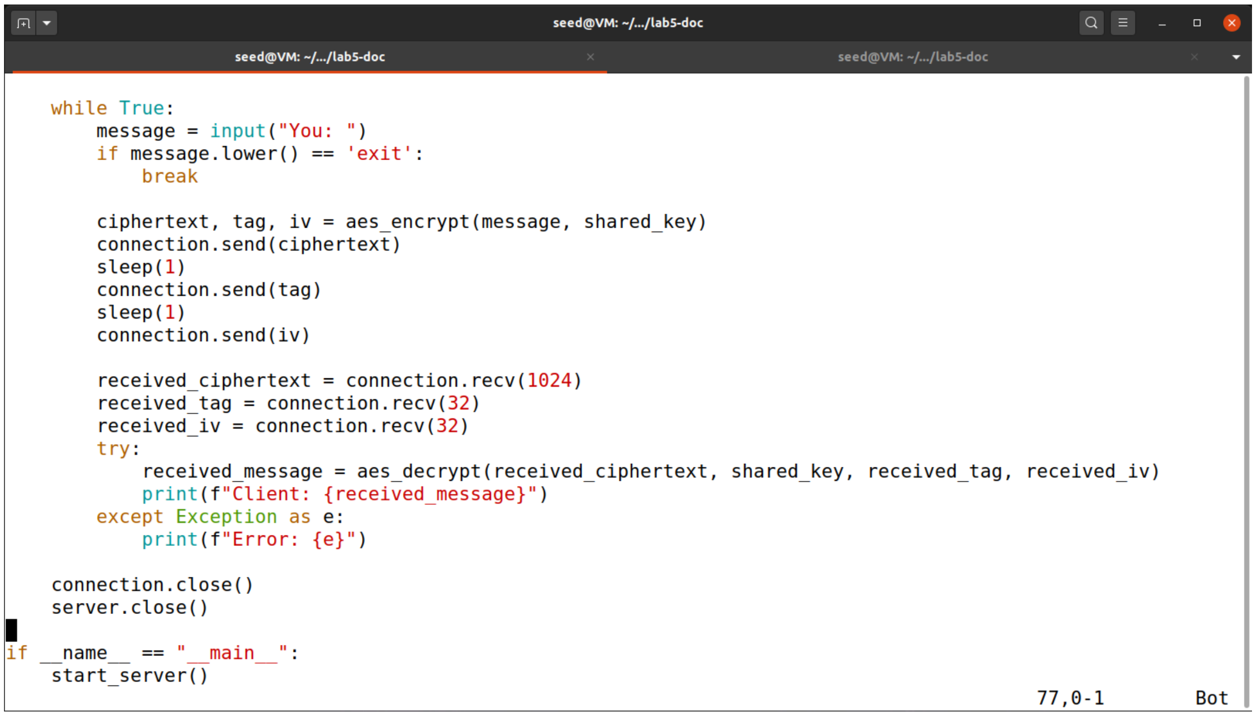
   server.py



```
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 = 258224987808690858965591917200301187432970579282922351283065935654064762201684119462964535328013783143590317197274755 9779
    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
```

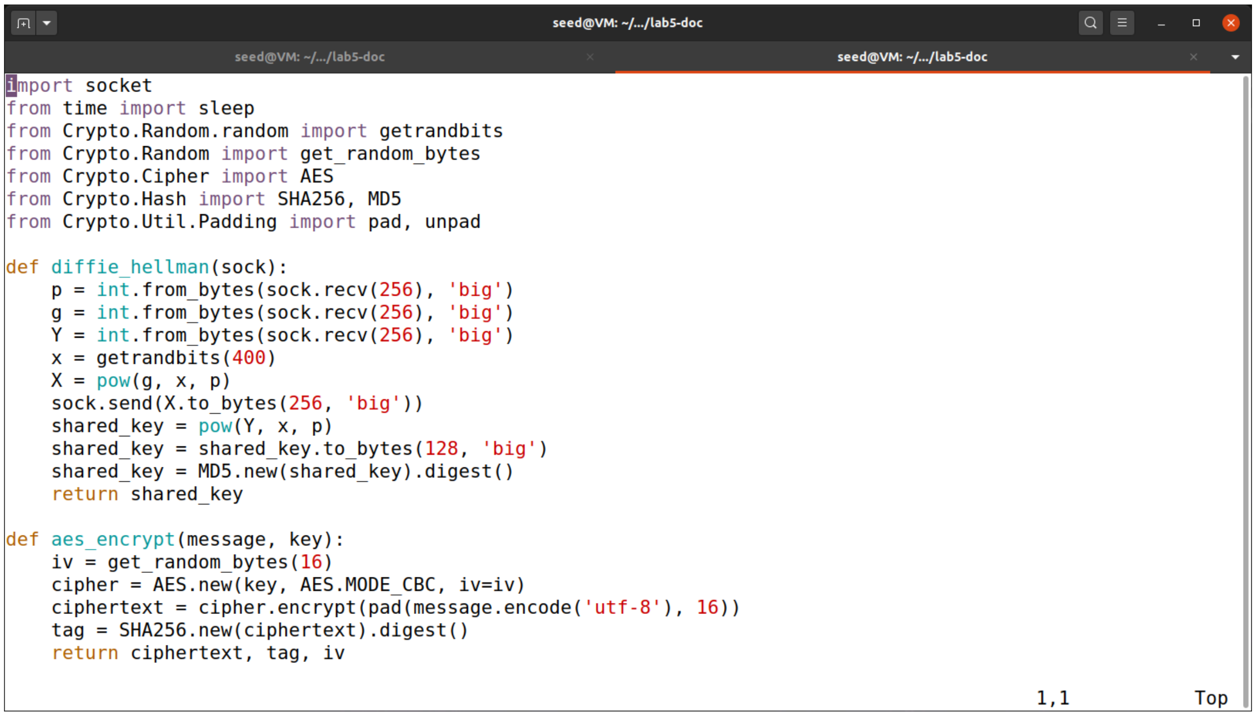


```
def aes_encrypt(message, key):
    iv = get_random_bytes(16)
    cipher = AES.new(key, AES.MODE_CBC, iv=iv)
    ciphertext = cipher.encrypt(pad(message.encode('utf-8'), 16))
    tag = SHA256.new(ciphertext).digest()
    return ciphertext, tag, iv

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%
```

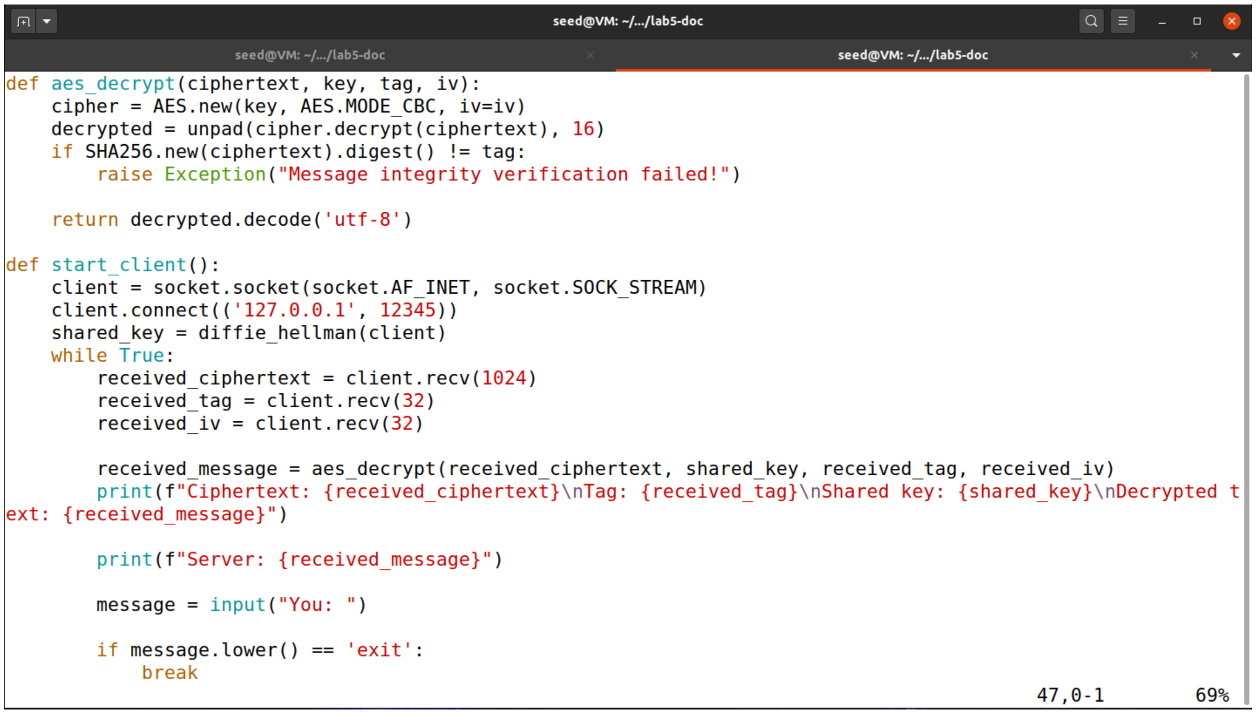


```

    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
```

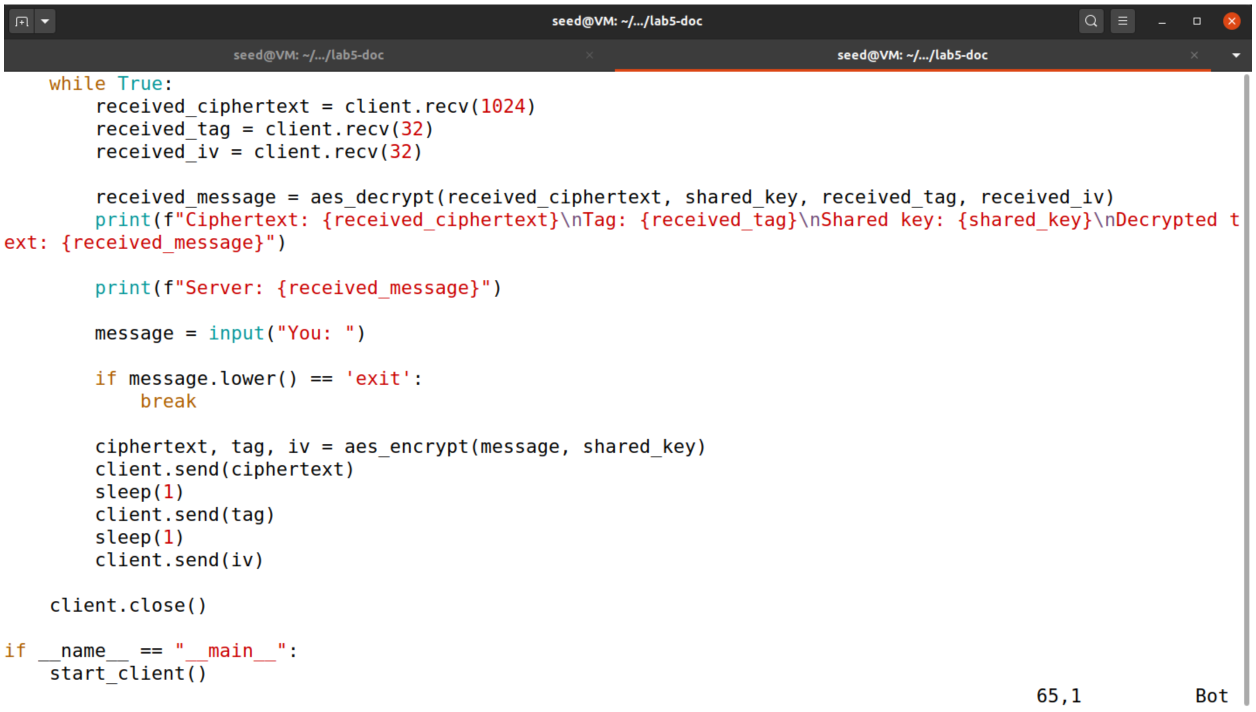
client.py



```
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
```

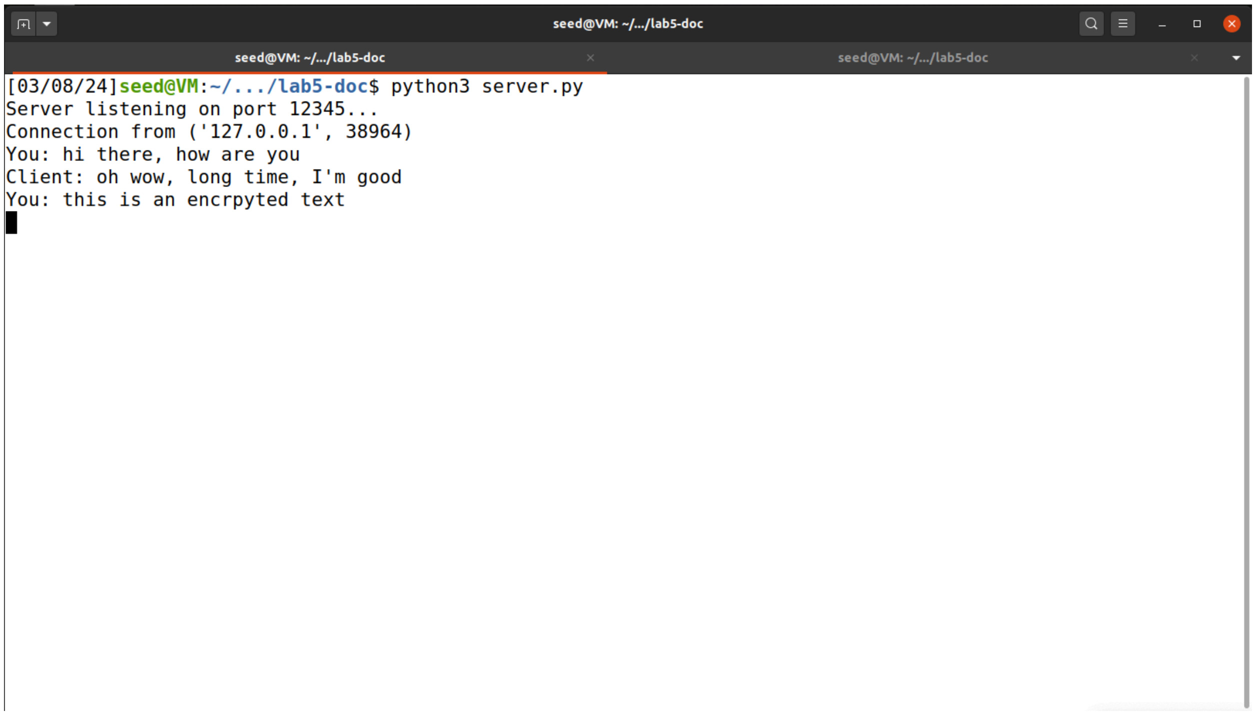


```
def aes_decrypt(ciphertext, key, tag, iv):
    cipher = AES.new(key, AES.MODE_CBC, iv=iv)
    decrypted = unpad(cipher.decrypt(ciphertext), 16)
    if SHA256.new(ciphertext).digest() != tag:
        raise Exception("Message integrity verification failed!")

    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%
```

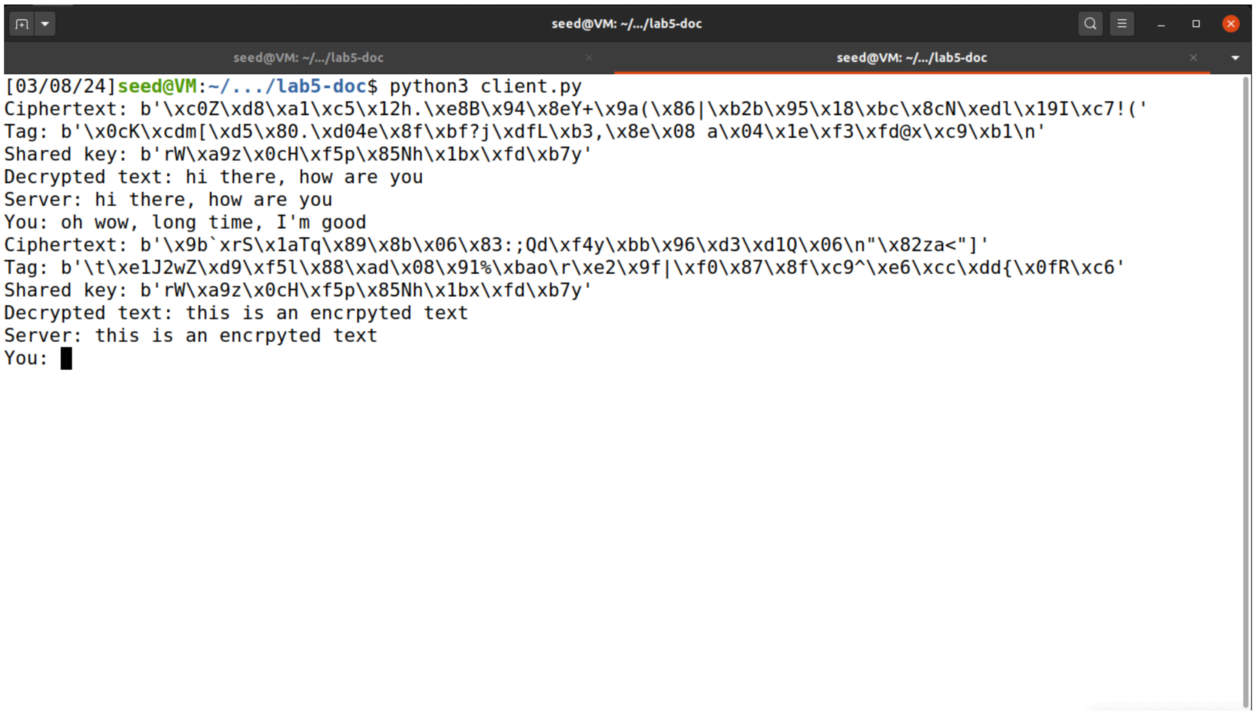


```
    while True:
        received_ciphertext = client.recv(1024)
        received_tag = client.recv(32)
        received_iv = client.recv(32)

        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: 
```
]