CPSC 352 Final Project

By Kush Patel, Daylan Stoica, Ryan Phillips, Mason Chiang, and Edmarck Sosa

This project is a secure file sharing system in which there are clients and a server, the server accepts multiple clients at a time, and when a client wants to share a file, the client sends the file to the server and any other clients listening to the server, and the other clients receive the file in which the server broadcasts the message containing the file, and the server displays the contents of the file

The services the project provides are encryption, hashing, confidentiality, and digital signature in which before sending, the client encrypts the file, the other client who is receiving the encrypted message decrypts the message, which in turn provides confidentiality, and the client who sends the message provides a digital signature through hashing and checks to see if it is valid and if it is valid it'll be decrypted, but if it is not valid it wont be encrypted. The digital signatures provide authentication, in which the client or server who is receiving the file message checks to see if the signature is valid and if it isn't valid then the message including the contents of the file wont be decrypted.

The design of the project was a programming project which is a software or program that was written in the Python programming langauge, it uses multiple python libraries and imports, to have this program running, you must first run the server with the command " python3 server.py 3456 127.0.0.1", if you try to run a client file first it will throw an exception message saying client doesn't exist, the components of this program or software are that its a server file, which accepts multiple client connections and receives message from a client, displays the message contents after decrypting it and broadcasts the messages containing the file contents to the other clients listening or waiting in the server, but doesn't send the message containing the file contents back to the client who sent it recently, instead it keeps it and waits for any more clients in the server, whenever a client disconnects from the server it sends a message saying client disconnected from the server and removes the client who disconnected from its list, if a new client joins the server, it adds the client to its list of clients, if it has problems accepting connections, it displays a message error accepting connection, the server accepts multiple clients at once. The client gets its messages sent to other clients why having the server broadcast the messages to all the clients except them. To run a client file, you have to use this command " python3 client.py 3456 127.0.0.1". The client can send or receive messages, the client first sends the message, but before doing it the message gets encrypted, then the message containing the file contents gets hashed

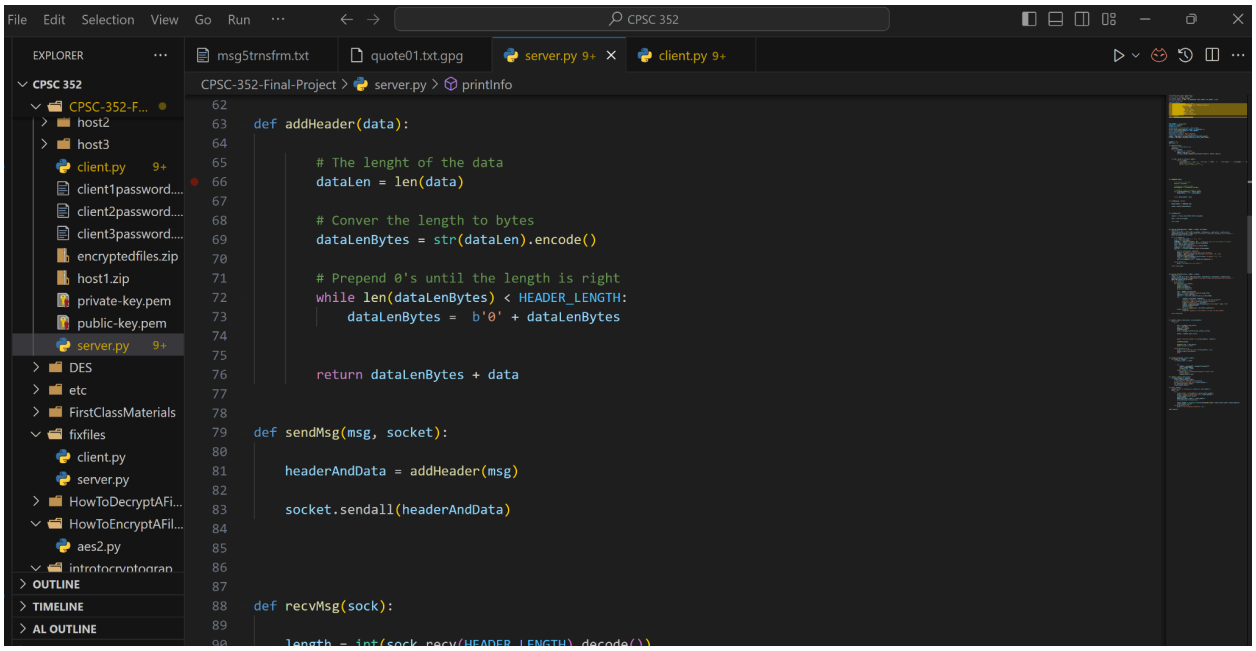
with a digital signature, the client inputs the host name which is the folder containing all the files and sends all the files and its contents in that folder to the server and the other clients.

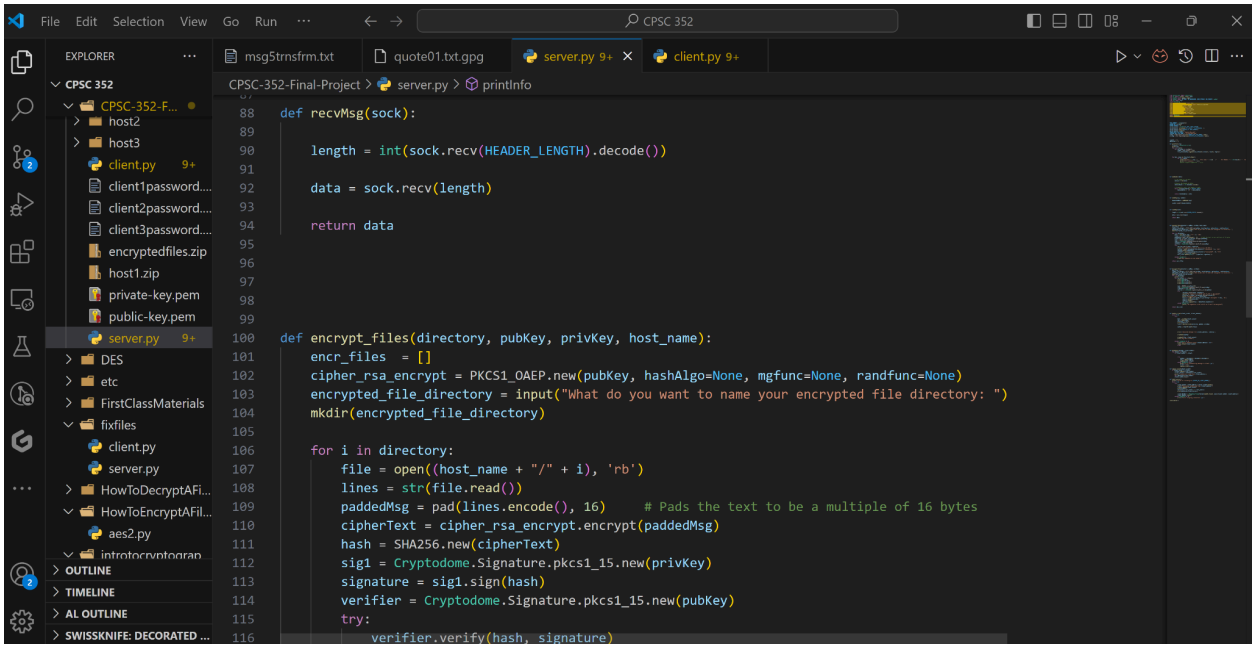
There are so many security protocols provided in this program, one of which is confidentiality in which the message containing all the files and its contents gets encrypted when it gets sent and gers decrypted when the client or server receives the message, the public and private keys of the program are then exchanged between the clients and server, they both use the same public and private key, which is symmetric key encryption in which it provides confidentiality, when the messages are being sent and before they are sent it goes through hashing in which a digital signature is created, in which it provides authentication and a digital signature between the clients and the server. It provides secure communications because the messages in which the file and its contents are being stored in are being encrypted first and hashed with a digital signature

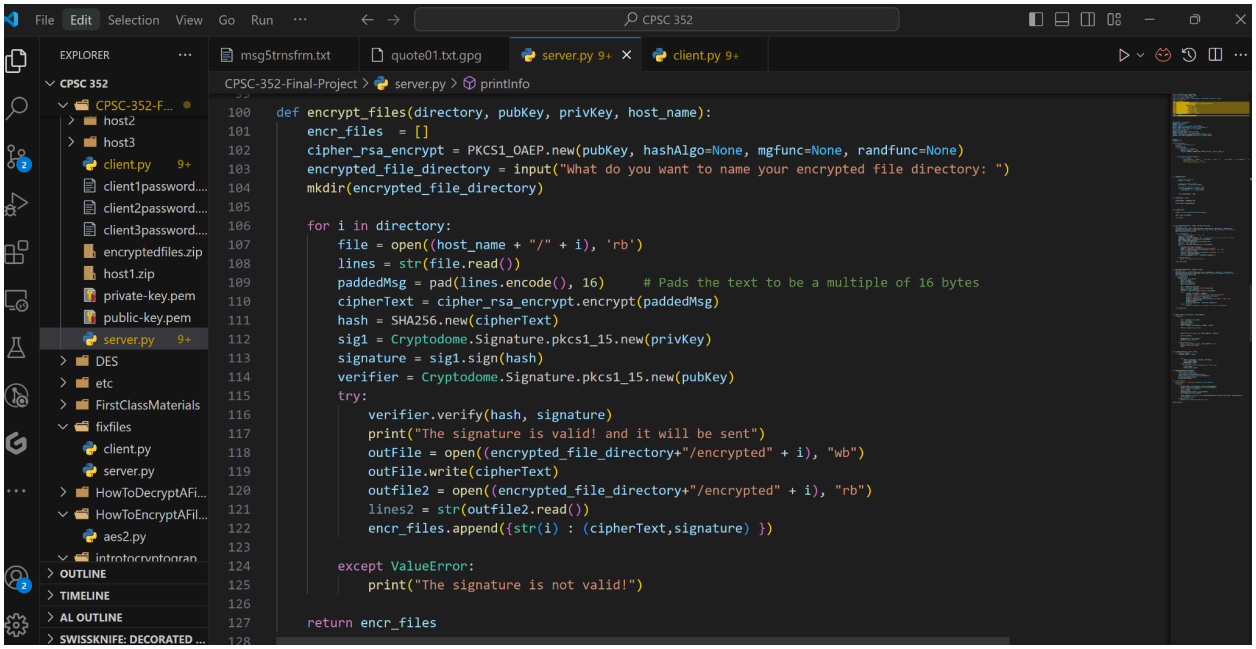
The code implementation was that it was written in the Python programming langauge, it uses multiple python libraries and imports like os, pickle, socket, sys,threading,Cryptodome,Crypto.Util ,and threading, to have this program running, you must first run the server with the command " python3 server.py 3456 127.0.0.1", if you try to run a client file first it will throw an exception message saying client doesn't exist. The server file has a "PORT_NUMBER" variable in which it gets the index 1 of what command we put in the terminal in which it is "3456", it also has a "SERVER_IP"variable in which it gets the index 2 of what command we put in the terminal in which it is "127.0.0.1" , they variables are set equal to argv[whatever index num], the argv is a function and attribute of the sys library, there is a variable called "HEADER_Length", which we use to unpickle the data structure that is being sent to the server that contains the message, file name, file, file contents, digital signature. There is a socket variable in which it creates a socket for receiving messages and it binds the server ip and the port number, the ip address and the port number for the clients must be the same as the server in order for it to connect successfully. The program uses the same public key and private key, in which in the program there are file names and the public and private key variables are then creating using a function called RSA.import_key which opens the files in which the two public and private keys are located in and gets the contents of the two files and stores them in a variable. There is a client list which has a list of all the clients that are connected and a addresses set that stores the set of all addresses of the clients connected, there is a printInfo function in which it takes a parameter in which you would put the data structure in and it creates a default dictionary with the list being the value and it looks through the second index of the data structure in which it contains the message containing the file and the file contents and it then appends the contents of the data structure to the file_dict variable which is the default dictionary, and the default

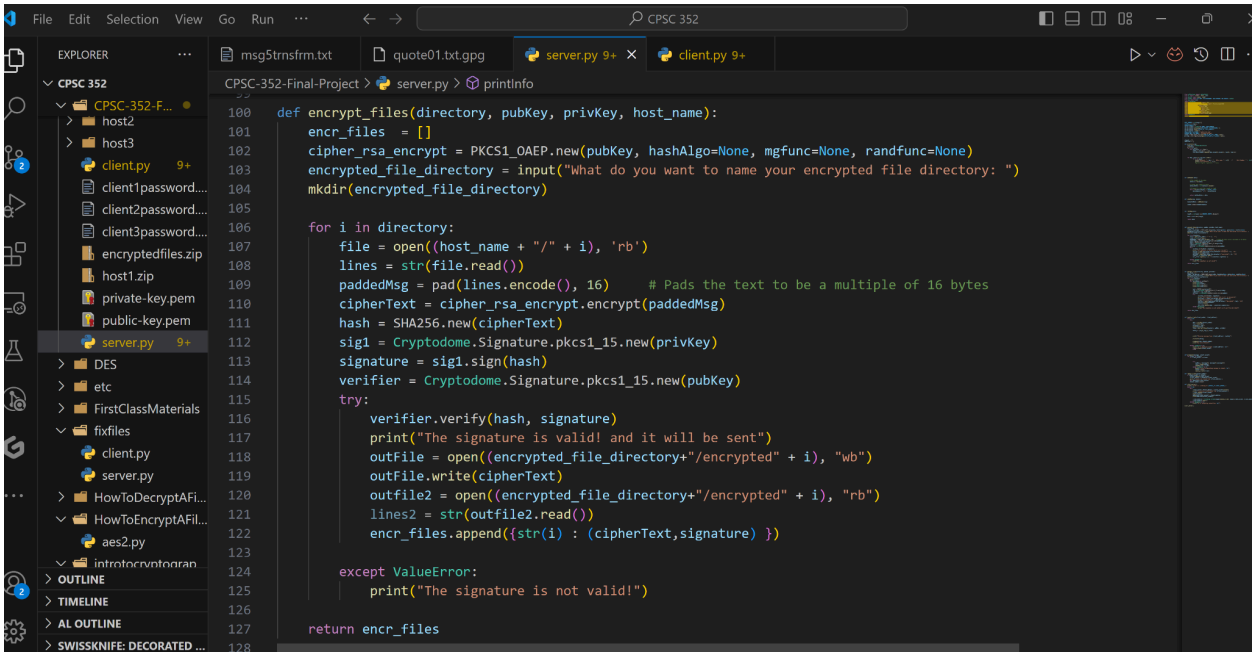
dictionary's key which is the "`newKey = key.split('.')[0]`", in which this variable gets all the text and characters before the '.' to keep the name of the file as its key and the value of it list of tuples in which each tuple contains the keyword, filename, port number, ip address, and file contents. There is an addHeader function which pickles the data structure that the server is trying to send from the original data structure to binary format it was being set as, there is a send message function which sends the pickled message to the other clients. The receive message unpickles the data structure or any contents being sent using the .decode() function and returns the unpickled information, there is an ecnrypt_function that creates a directory that stores all the encrypted files using the os module, it appends all the encrypted files in a list, it reads through the the files, pads the text to be a multiple of 16 bytes, it then hashes the contents of the message and file and gives them a digital signature, the encrypted files are appended to the list in a form of a diction with the file name being th key and the ciphertext which is the encrypted contents of the file,and the signature, if the signature isn't valid, the message and contents of the file won't be encrypted or sent, there is decrypt file function, which decrypts all the files, it creates a directory in which all the decrypted files are stored in and if the signature is valid the files will be decrypred if not then then files won't be decrypted, there is a handle_client function in which the server looks for clients currently connected and receives data from the clients, they continuously look for clients and accepts multiple clients through threading, there is broadcast function that sends the message sent from a recent client to all the other clients in the server, it looks through the list of clients and checks to see if it isn't the recent client who sent that message and then they send it to all the clients in the list, there is a remove client function that removes the client from the server if they disconnect and prints out where the connection closed from by accessing the addresses set that stores all the address of all the clients that disconnected, there is a start server function which continuously accepts clients and appends each client connected to the client list and gets the address of the client and stores it in the addresses dictionary, then it does another thread and calls the handle client function in the thread and because of the thread function multiple clients can be accepted at once and the file can be run at multiple times, threads cause the program to have two things happening at once and the server function runs all the functions. To run the client file you have to do " python3 client.py 3456 127.0.0.1", for it to run successfully both ip address and port number have to be the same. " python3 server.py 3456 127.0.0.1".There is an addHeader function which pickles the data structure that the server is trying to send from the original data structure to binary format it was being set as, there is a send message function which sends the pickled message to the other clients.There is an encrypt files functions in which there is an ecnrypt_function that creates a directory that stores all the encrypted files using the os module, it appends all the encrypted files in a list, it reads through the the files, pads the text to be a multiple of 16 bytes, it then hashes the contents of the message and file

and gives them a digital signature, the encrypted files are appended to the list in a form of a dictionary with the file name being th key and the ciphertext which is the encrypted contents of the file,and the signature, if the signature isn't valid, the message and contents of the file won't be encrypted or sent, there is decrypt file function, which decrypts all the files, it creates a directory in which all the decrypted files are stored in and if the signature is valid the files will be decrypred if not then then files won't be decrypted, there is a receive messages function in which theres a loop in which it continually looks for messages and it unpickles all the data in the data structure and it then decrypts the files in the function and it prints all the contents of the decrypted files,the main part of the client has a port number variable which looks at the command and takes the 1 index of the command using argv and uses the same public and private key to encrypt and decrypt files like the server does, it also uses threads to recieve messages similar to what the server does, the client receives messages similar to what the server does, but sends messages differently, in which the client only sends to the server , the server sends to multiple clients.

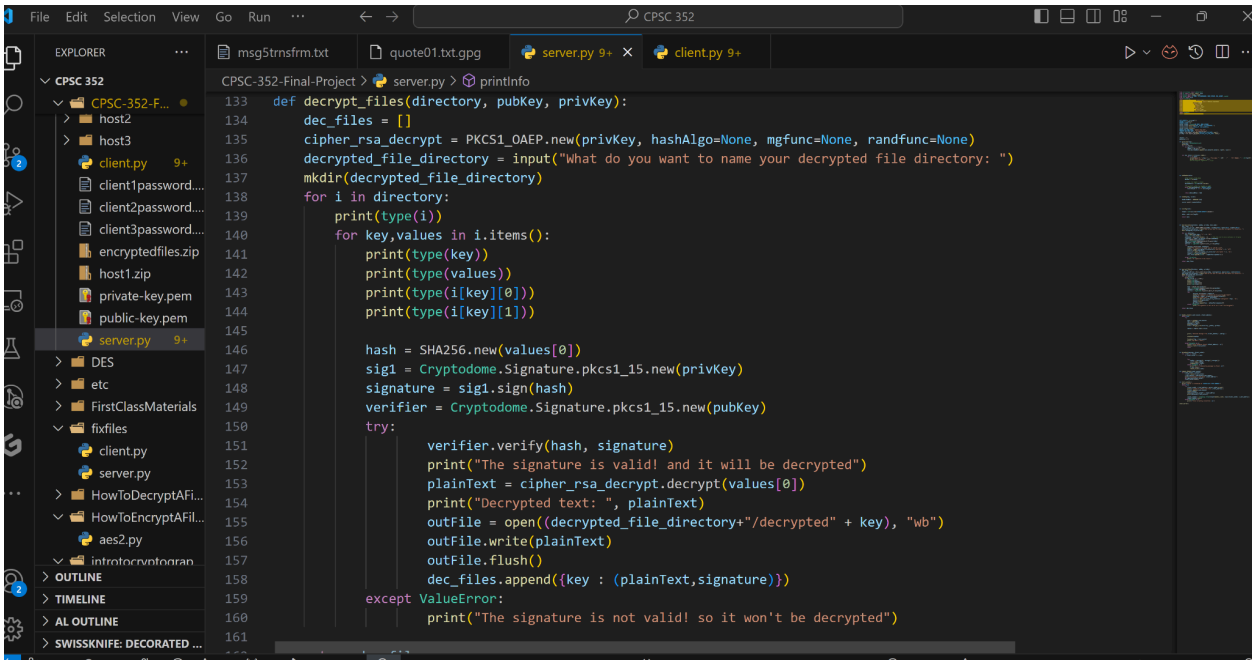



File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > server.py > printInfo
def recvMsg(sock):
length = int(sock.recv(HEADER_LENGTH).decode())
data = sock.recv(length)
return data
def encrypt_files(directory, pubKey, privKey, host_name):
encr_files = []
cipher_rsa_encrypt = PKCS1_OAEP.new(pubKey, hashAlgo=None, mgfunc=None, randfunc=None)
encrypted_file_directory = input("What do you want to name your encrypted file directory: ")
mkdir(encrypted_file_directory)
for i in directory:
file = open((host_name + "/" + i), 'rb')
lines = str(file.read())
paddedMsg = pad(lines.encode(), 16) # Pads the text to be a multiple of 16 bytes
cipherText = cipher_rsa_encrypt.encrypt(paddedMsg)
hash = SHA256.new(cipherText)
sig1 = Cryptodome.Signature.pkcs1_15.new(privKey)
signature = sig1.sign(hash)
verifier = Cryptodome.Signature.pkcs1_15.new(pubKey)
try:
verifier.verify(hash, signature)
CPSC 352
host2
host3
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
host1.zip
private-key.pem
public-key.pem
server.py 9+
DES
etc
FirstClassMaterials
fixfiles
client.py
server.py
HowToDecryptAFi...
HowToEncryptAFil...
aes2.py
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...


File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > server.py > printInfo
def encrypt_files(directory, pubKey, privKey, host_name):
encr_files = []
cipher_rsa_encrypt = PKCS1_OAEP.new(pubKey, hashAlgo=None, mgfunc=None, randfunc=None)
encrypted_file_directory = input("What do you want to name your encrypted file directory: ")
mkdir(encrypted_file_directory)
for i in directory:
file = open((host_name + "/" + i), 'rb')
lines = str(file.read())
paddedMsg = pad(lines.encode(), 16) # Pads the text to be a multiple of 16 bytes
cipherText = cipher_rsa_encrypt.encrypt(paddedMsg)
hash = SHA256.new(cipherText)
sig1 = Cryptodome.Signature.pkcs1_15.new(privKey)
signature = sig1.sign(hash)
verifier = Cryptodome.Signature.pkcs1_15.new(pubKey)
try:
verifier.verify(hash, signature)
print("The signature is valid! and it will be sent")
outFile = open((encrypted_file_directory+"/encrypted" + i), "wb")
outFile.write(cipherText)
outfile2 = open((encrypted_file_directory+"/encrypted" + i), "rb")
lines2 = str(outfile2.read())
encr_files.append({str(i) : (cipherText,signature) })
except ValueError:
print("The signature is not valid!")
return encr_files
CPSC 352
host2
host3
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
host1.zip
private-key.pem
public-key.pem
server.py 9+
DES
etc
FirstClassMaterials
fixfiles
client.py
server.py
HowToDecryptAFi...
HowToEncryptAFil...
aes2.py
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...

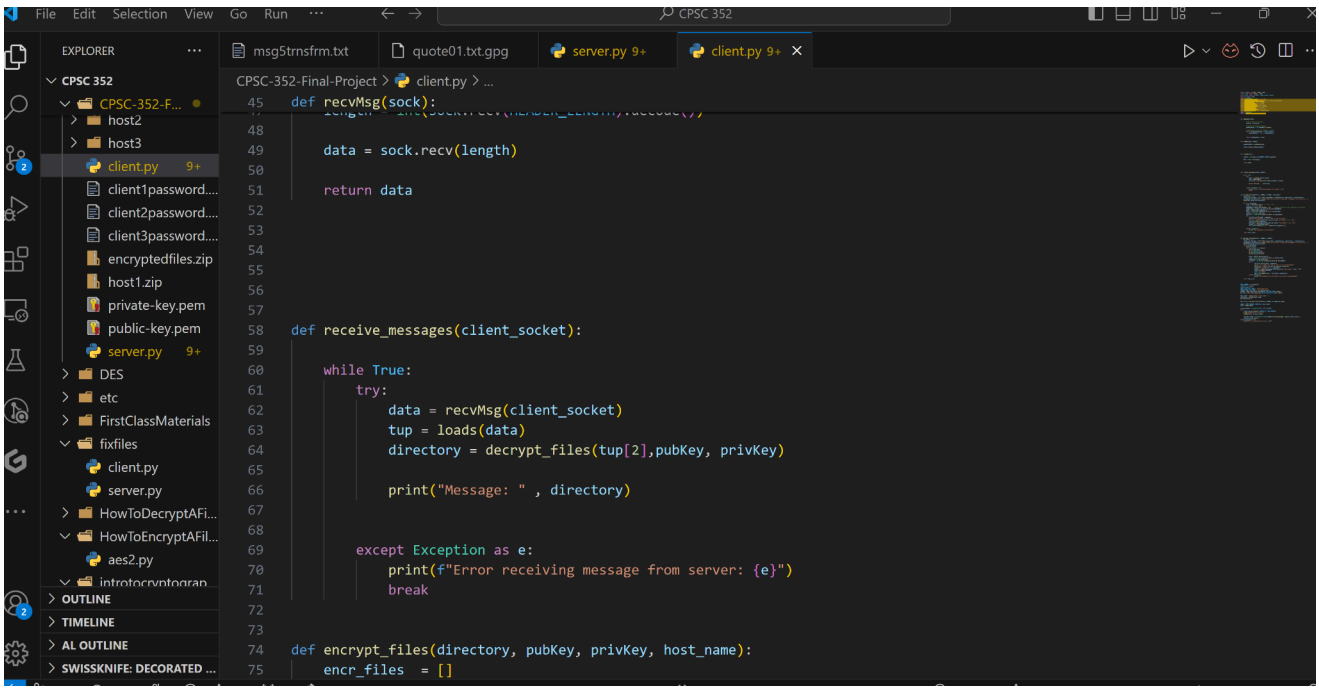


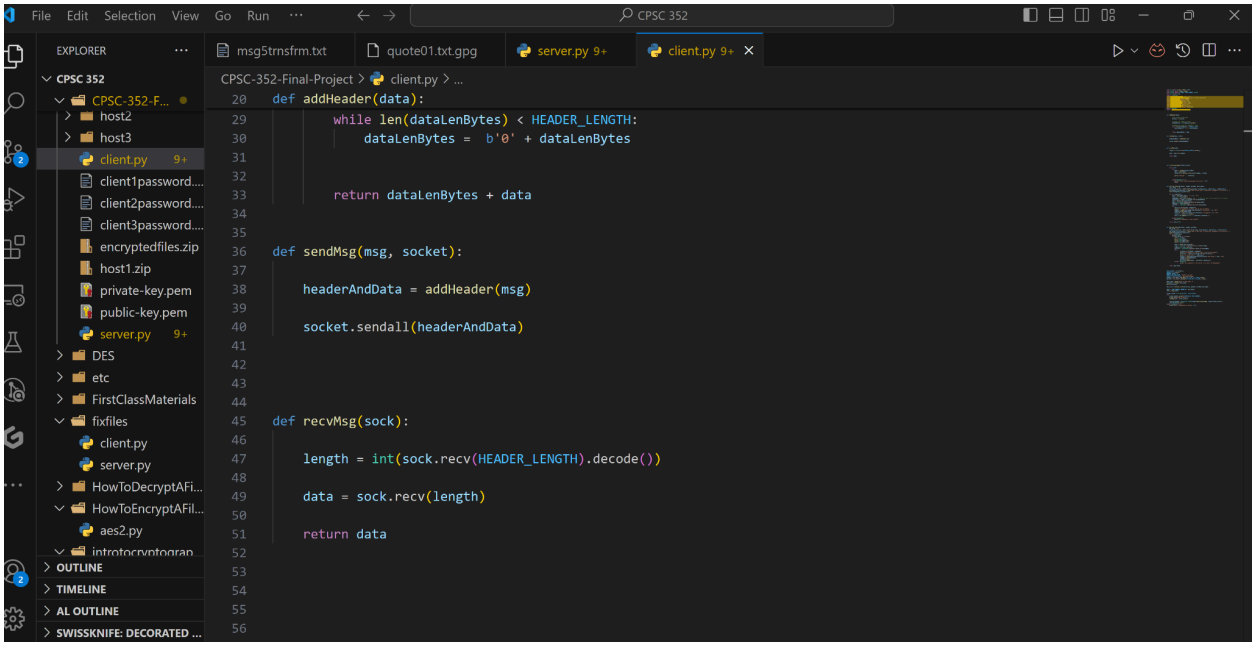
```python
def encrypt_files(directory, pubKey, privKey, host_name):
    encr_files  = []
    cipher_rsa_encrypt = PKCS1_OAEP.new(pubKey, hashAlgo=None, mgfunc=None, randfunc=None)
    encrypted_file_directory = input("What do you want to name your encrypted file directory: ")
    mkdir(encrypted_file_directory)

    for i in directory:
        file = open((host_name + "/" + i), 'rb')
        lines = str(file.read())
        paddedMsg = pad(lines.encode(), 16)     # Pads the text to be a multiple of 16 bytes
        cipherText = cipher_rsa_encrypt.encrypt(paddedMsg)
        hash = SHA256.new(cipherText)
        sig1 = Cryptodome.Signature.pkcs1_15.new(privKey)
        signature = sig1.sign(hash)
        verifier = Cryptodome.Signature.pkcs1_15.new(pubKey)
        try:
            verifier.verify(hash, signature)
            print("The signature is valid! and it will be sent")
            outFile = open((encrypted_file_directory+"/encrypted" + i), "wb")
            outFile.write(cipherText)
            outfile2 = open((encrypted_file_directory+"/encrypted" + i), "rb")
            lines2 = str(outfile2.read())
            encr_files.append({str(i) : (cipherText,signature) })

        except ValueError:
            print("The signature is not valid!")

    return encr_files
```

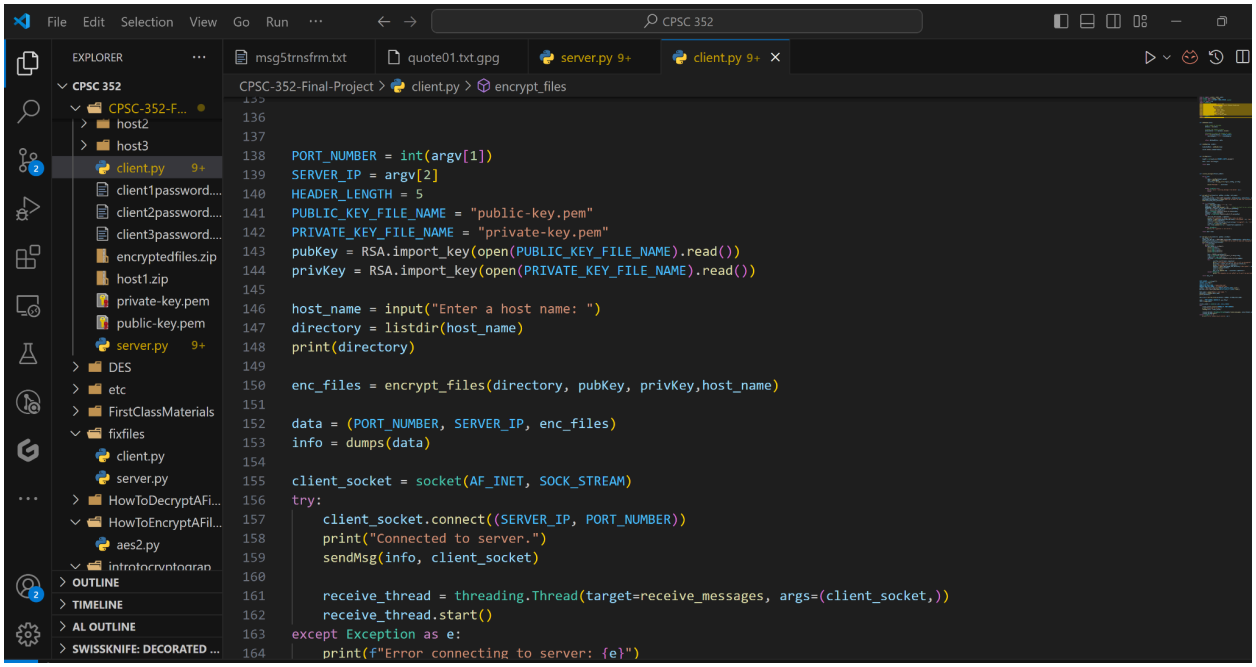


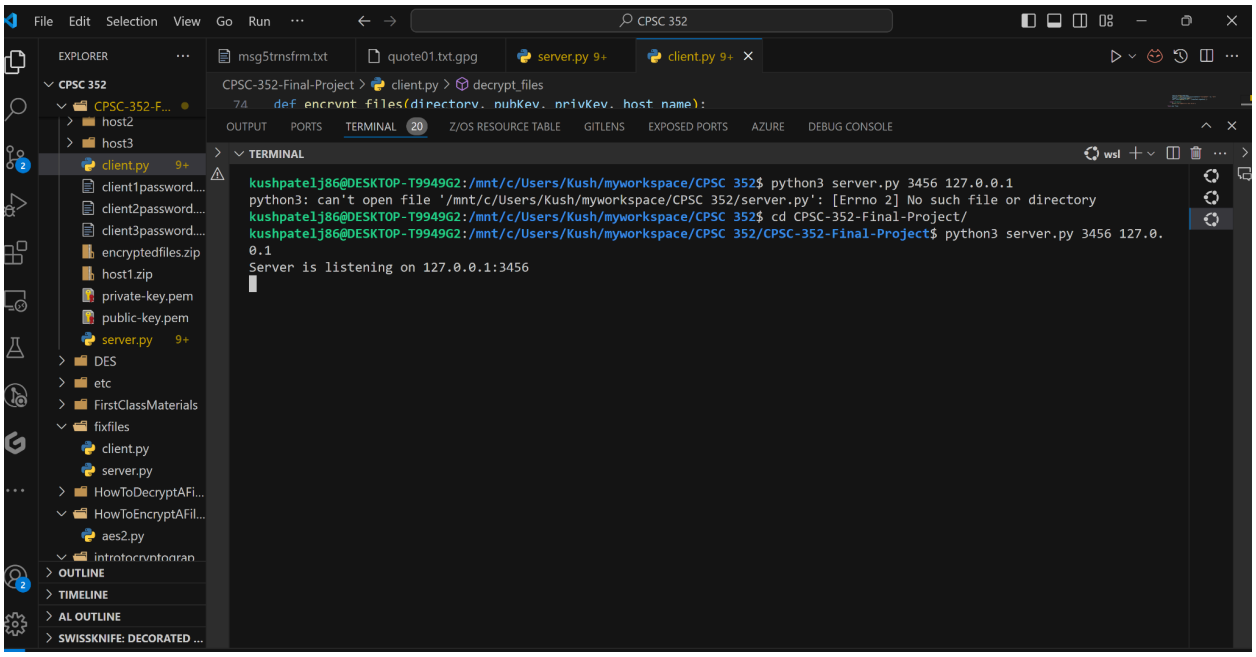
```python
def decrypt_files(directory, pubKey, privKey):
    dec_files = []
    cipher_rsa_decrypt = PKCS1_OAEP.new(privKey, hashAlgo=None, mgfunc=None, randfunc=None)
    decrypted_file_directory = input("What do you want to name your decrypted file directory: ")
    mkdir(decrypted_file_directory)
    for i in directory:
        print(type(i))
        for key,values in i.items():
            print(type(key))
            print(type(values))
            print(type(i[key][0]))
            print(type(i[key][1]))

            hash = SHA256.new(values[0])
            sig1 = Cryptodome.Signature.pkcs1_15.new(privKey)
            signature = sig1.sign(hash)
            verifier = Cryptodome.Signature.pkcs1_15.new(pubKey)
            try:
                    verifier.verify(hash, signature)
                    print("The signature is valid! and it will be decrypted")
                    plainText = cipher_rsa_decrypt.decrypt(values[0])
                    print("Decrypted text: ", plainText)
                    outFile = open((decrypted_file_directory+"/decrypted" + key), "wb")
                    outFile.write(plainText)
                    outFile.flush()
                    dec_files.append({key : (plainText,signature)})
            except ValueError:
                    print("The signature is not valid! so it won't be decrypted")
```

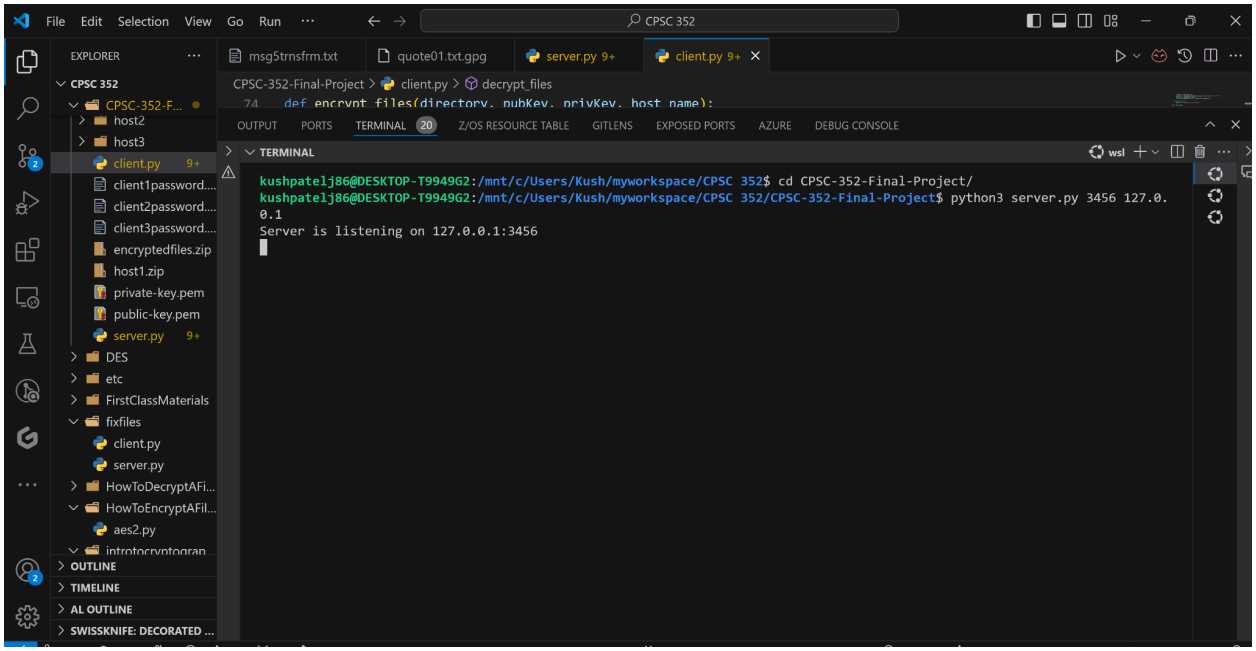
```python
def encrypt_files(directory, pubKey, privKey, host_name):
    encr_files  = []
    cipher_rsa_encrypt = PKCS1_OAEP.new(pubKey, hashAlgo=None, mgfunc=None, randfunc=None)
    encrypted_file_directory = input("What do you want to name your encrypted file directory: ")
    mkdir(encrypted_file_directory)

    for i in directory:
        file = open((host_name + "/" + i), 'rb')
        lines = str(file.read())
        paddedMsg = pad(lines.encode(), 16)     # Pads the text to be a multiple of 16 bytes
        cipherText = cipher_rsa_encrypt.encrypt(paddedMsg)
        hash = SHA256.new(cipherText)
        sig1 = Cryptodome.Signature.pkcs1_15.new(privKey)
        signature = sig1.sign(hash)
        verifier = Cryptodome.Signature.pkcs1_15.new(pubKey)
        try:
            verifier.verify(hash, signature)
            print("The signature is valid! and it will be sent")
            outFile = open((encrypted_file_directory+"/encrypted" + i), "wb")
            outFile.write(cipherText)
            outfile2 = open((encrypted_file_directory+"/encrypted" + i), "rb")
            lines2 = str(outfile2.read())
            encr_files.append({str(i) : (cipherText,signature) })

        except ValueError:
            print("The signature is not valid!")

    return encr_files
```

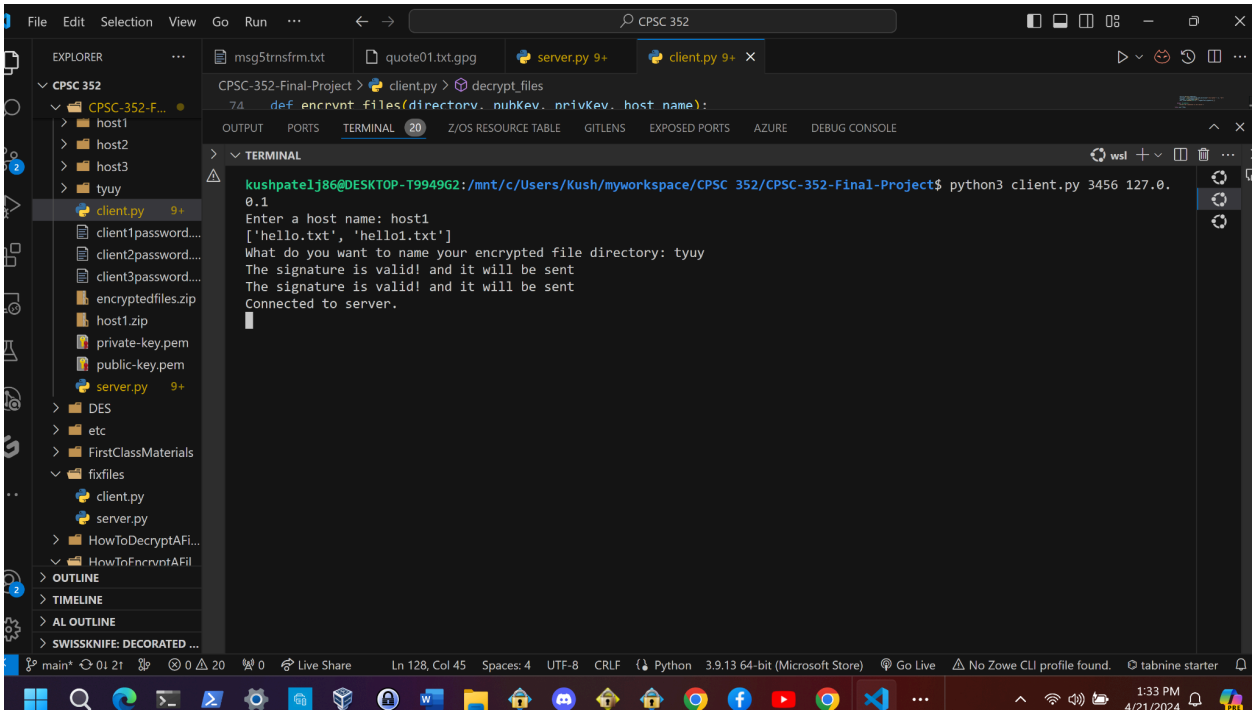


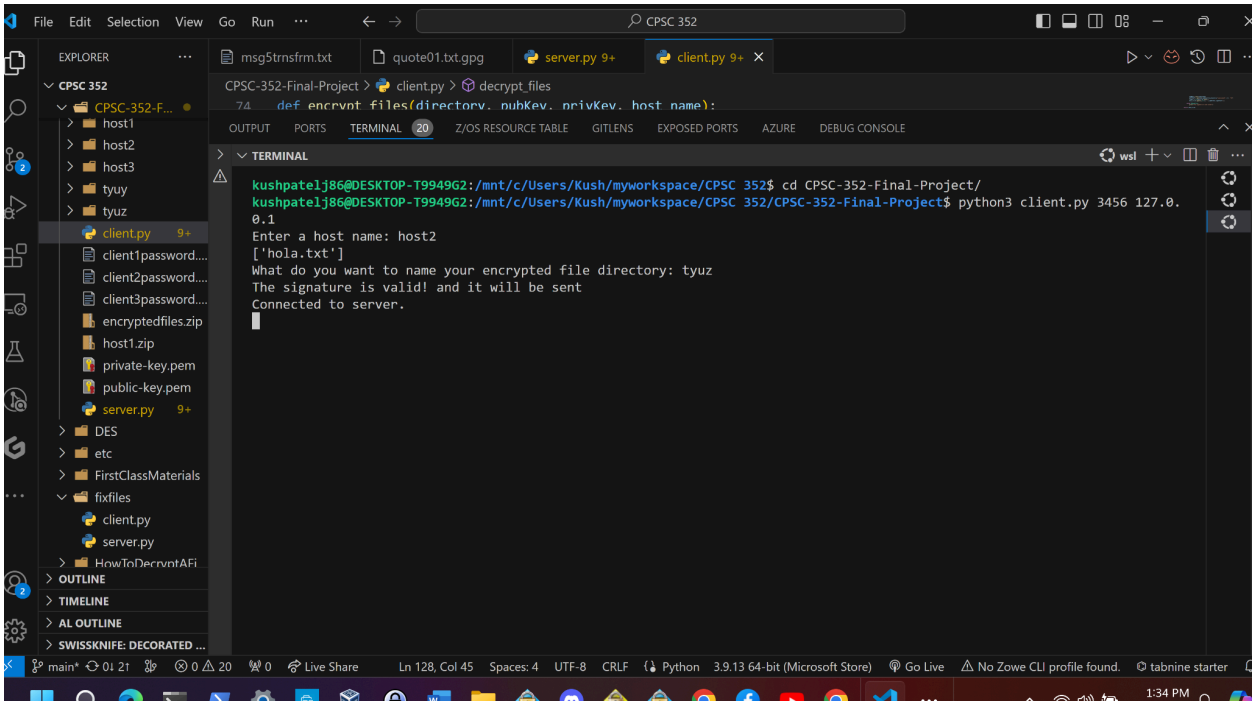

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > client.py > ...
def addHeader(data):
while len(dataLenBytes) < HEADER_LENGTH:
dataLenBytes = b'0' + dataLenBytes
return dataLenBytes + data
def sendMsg(msg, socket):
headerAndData = addHeader(msg)
socket.sendall(headerAndData)
def recvMsg(sock):
length = int(sock.recv(HEADER_LENGTH).decode())
data = sock.recv(length)
return data
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...

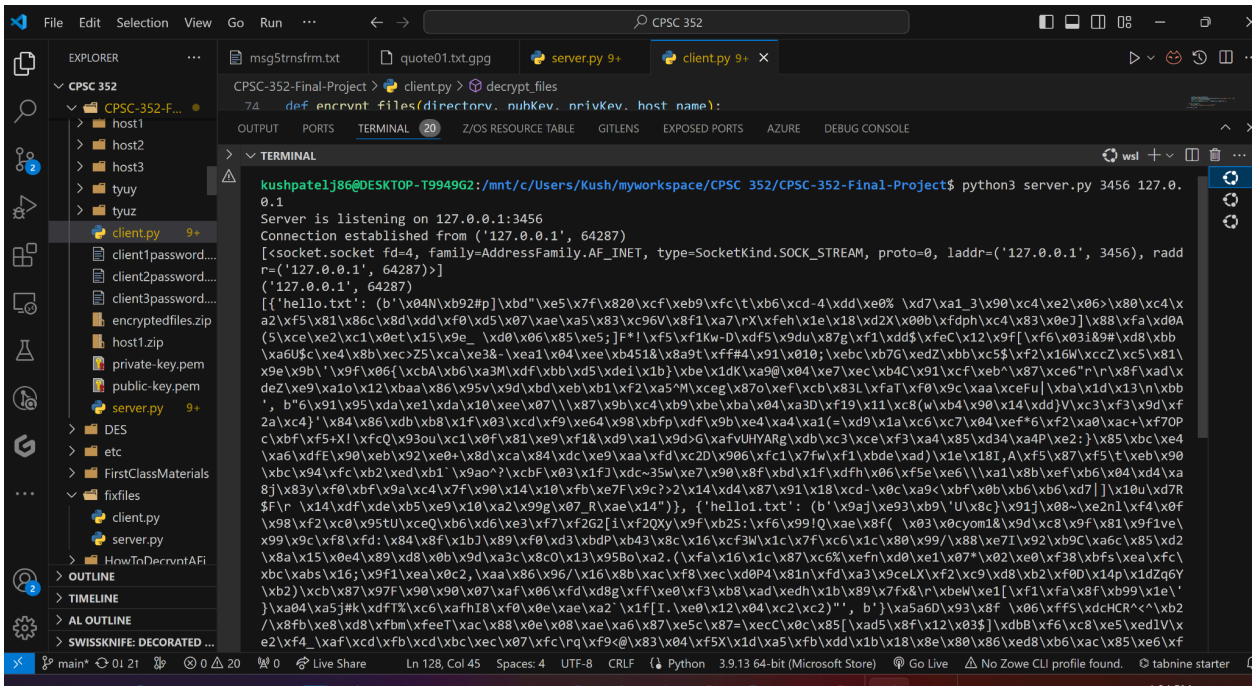

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > client.py > encrypt_files
PORT_NUMBER = int(argv[1])
SERVER_IP = argv[2]
HEADER_LENGTH = 5
PUBLIC_KEY_FILE_NAME = "public-key.pem"
PRIVATE_KEY_FILE_NAME = "private-key.pem"
pubKey = RSA.import_key(open(PUBLIC_KEY_FILE_NAME).read())
privKey = RSA.import_key(open(PRIVATE_KEY_FILE_NAME).read())
host_name = input("Enter a host name: ")
directory = listdir(host_name)
print(directory)
enc_files = encrypt_files(directory, pubKey, privKey,host_name)
data = (PORT_NUMBER, SERVER_IP, enc_files)
info = dumps(data)
client_socket = socket(AF_INET, SOCK_STREAM)
try:
client_socket.connect((SERVER_IP, PORT_NUMBER))
print("Connected to server.")
sendMsg(info, client_socket)
receive_thread = threading.Thread(target=receive_messages, args=(client_socket,))
receive_thread.start()
except Exception as e:
print(f"Error connecting to server: {e}")
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...

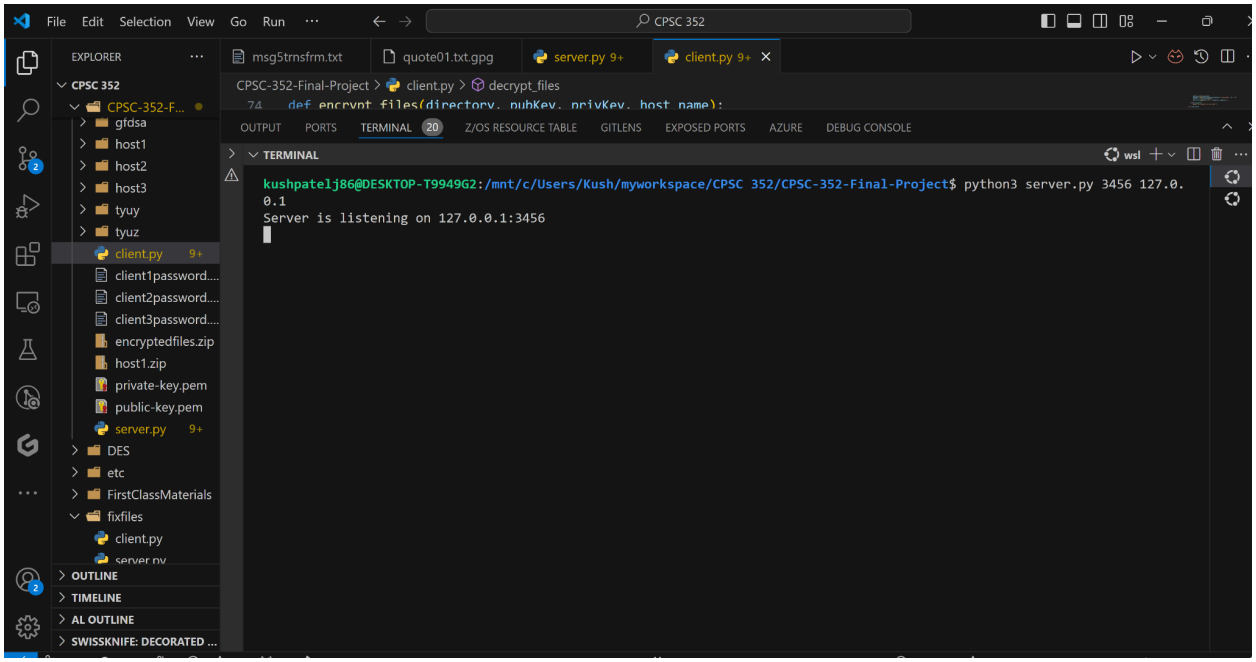

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > client.py > decrypt_files
OUTPUT PORTS TERMINAL 20 Z/OS RESOURCE TABLE GITLENS EXPOSED PORTS AZURE DEBUG CONSOLE
TERMINAL
kushpatelj86@DESKTOP-T9949G2:/mnt/c/Users/Kush/myworkspace/CPSC 352$ python3 server.py 3456 127.0.0.1
python3: can't open file '/mnt/c/Users/Kush/myworkspace/CPSC 352/server.py': [Errno 2] No such file or directory
kushpatelj86@DESKTOP-T9949G2:/mnt/c/Users/Kush/myworkspace/CPSC 352$ cd CPSC-352-Final-Project/
kushpatelj86@DESKTOP-T9949G2:/mnt/c/Users/Kush/myworkspace/CPSC 352/CPSC-352-Final-Project$ python3 server.py 3456 127.0.0.1
Server is listening on 127.0.0.1:3456
host2
host3
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
host1.zip
private-key.pem
public-key.pem
server.py 9+
DES
etc
FirstClassMaterials
fixfiles
client.py
server.py
HowToDecryptAFi...
HowToEncryptAFil...
aes2.py
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...

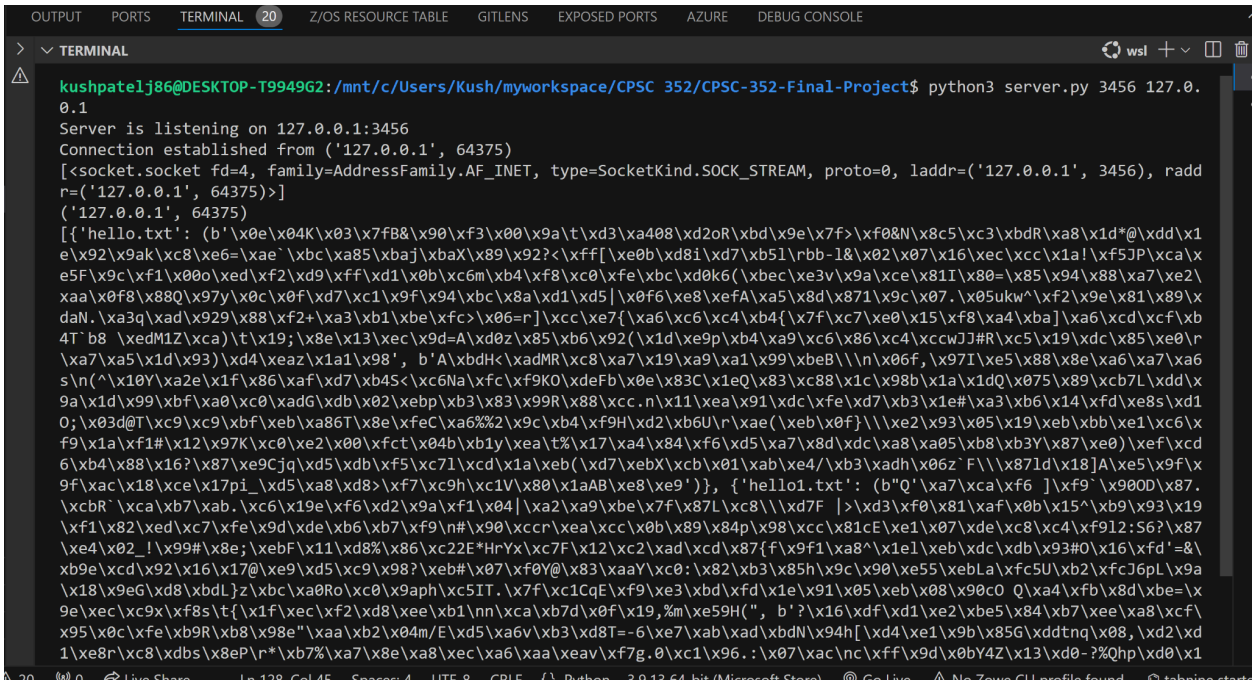

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > client.py > decrypt_files
OUTPUT PORTS TERMINAL 20 Z/OS RESOURCE TABLE GITLENS EXPOSED PORTS AZURE DEBUG CONSOLE
TERMINAL
kushpatelj86@DESKTOP-T9949G2:/mnt/c/Users/Kush/myworkspace/CPSC 352$ cd CPSC-352-Final-Project/
kushpatelj86@DESKTOP-T9949G2:/mnt/c/Users/Kush/myworkspace/CPSC 352/CPSC-352-Final-Project$ python3 server.py 3456 127.0.0.1
Server is listening on 127.0.0.1:3456
host2
host3
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
host1.zip
private-key.pem
public-key.pem
server.py 9+
DES
etc
FirstClassMaterials
fixfiles
client.py
server.py
HowToDecryptAFi...
HowToEncryptAFil...
aes2.py
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...

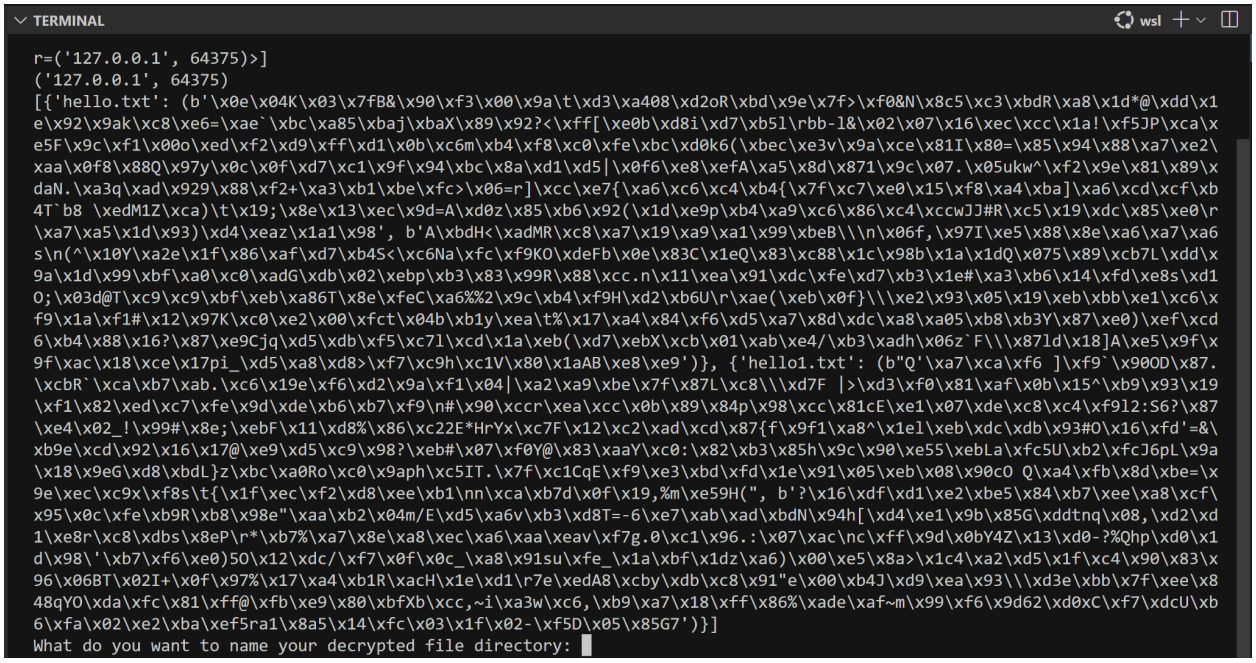

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > client.py > decrypt_files
def encrypt_files(directory, pubKey, privKey, host_name):
OUTPUT PORTS TERMINAL 20 Z/OS RESOURCE TABLE GITLENS EXPOSED PORTS AZURE DEBUG CONSOLE
TERMINAL
kushpatelj86@DESKTOP-T9949G2:/mnt/c/Users/Kush/myworkspace/CPSC 352/CPSC-352-Final-Project$ python3 client.py 3456 127.0.0.1
Enter a host name: host1
['hello.txt', 'hello1.txt']
What do you want to name your encrypted file directory: tyuy
The signature is valid! and it will be sent
The signature is valid! and it will be sent
Connected to server.
host1
host2
host3
tyuy
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
host1.zip
private-key.pem
public-key.pem
server.py 9+
DES
etc
FirstClassMaterials
fixfiles
client.py
server.py
HowToDecryptAFi...
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...
Ln 128, Col 45 Spaces: 4 UTF-8 CRLF Python 3.9.13 64-bit (Microsoft Store) Go Live No Zowe CLI profile found. tabnine starter
1:33 PM
4/21/2024

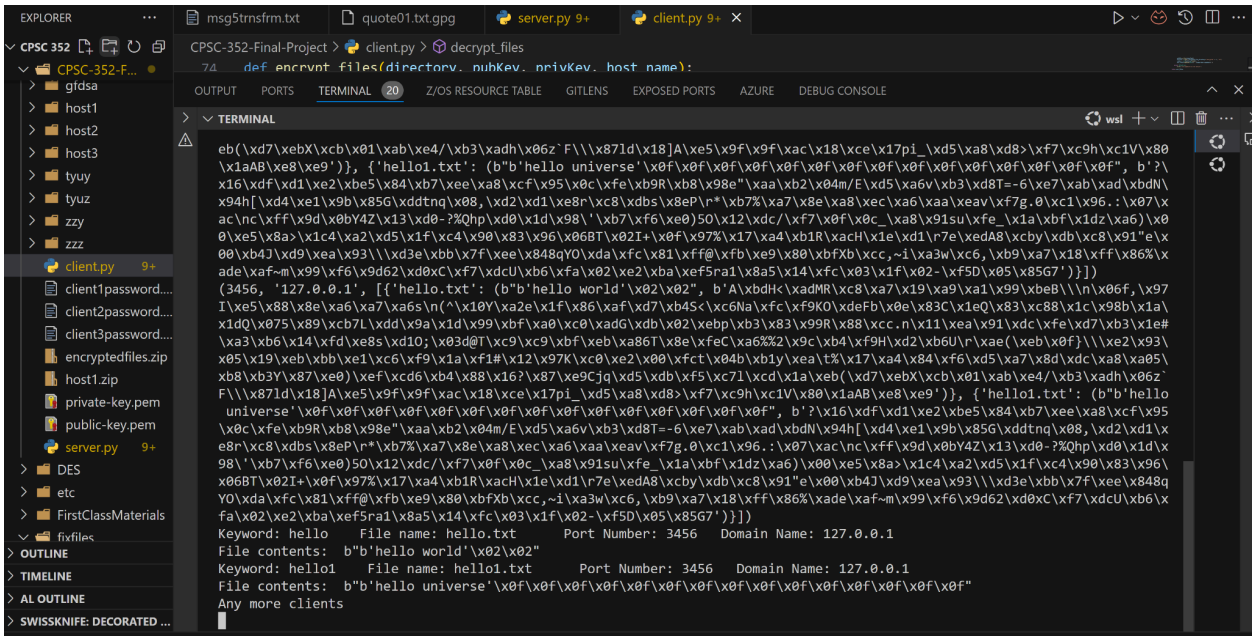

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > client.py > decrypt_files
def encrypt_files(directory, pubKey, privKey, host_name):
OUTPUT PORTS TERMINAL 20 Z/OS RESOURCE TABLE GITLENS EXPOSED PORTS AZURE DEBUG CONSOLE
TERMINAL
kushpatelj86@DESKTOP-T9949G2:/mnt/c/Users/Kush/myworkspace/CPSC 352$ cd CPSC-352-Final-Project/
kushpatelj86@DESKTOP-T9949G2:/mnt/c/Users/Kush/myworkspace/CPSC 352/CPSC-352-Final-Project$ python3 client.py 3456 127.0.0.1
Enter a host name: host2
['hola.txt']
What do you want to name your encrypted file directory: tyuz
The signature is valid! and it will be sent
Connected to server.
host1
host2
host3
tyuy
tyuz
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
host1.zip
private-key.pem
public-key.pem
server.py 9+
DES
etc
FirstClassMaterials
fixfiles
client.py
server.py
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...
Ln 128, Col 45 Spaces: 4 UTF-8 CRLF Python 3.9.13 64-bit (Microsoft Store) Go Live No Zowe CLI profile found. tabnine starter
1:34 PM

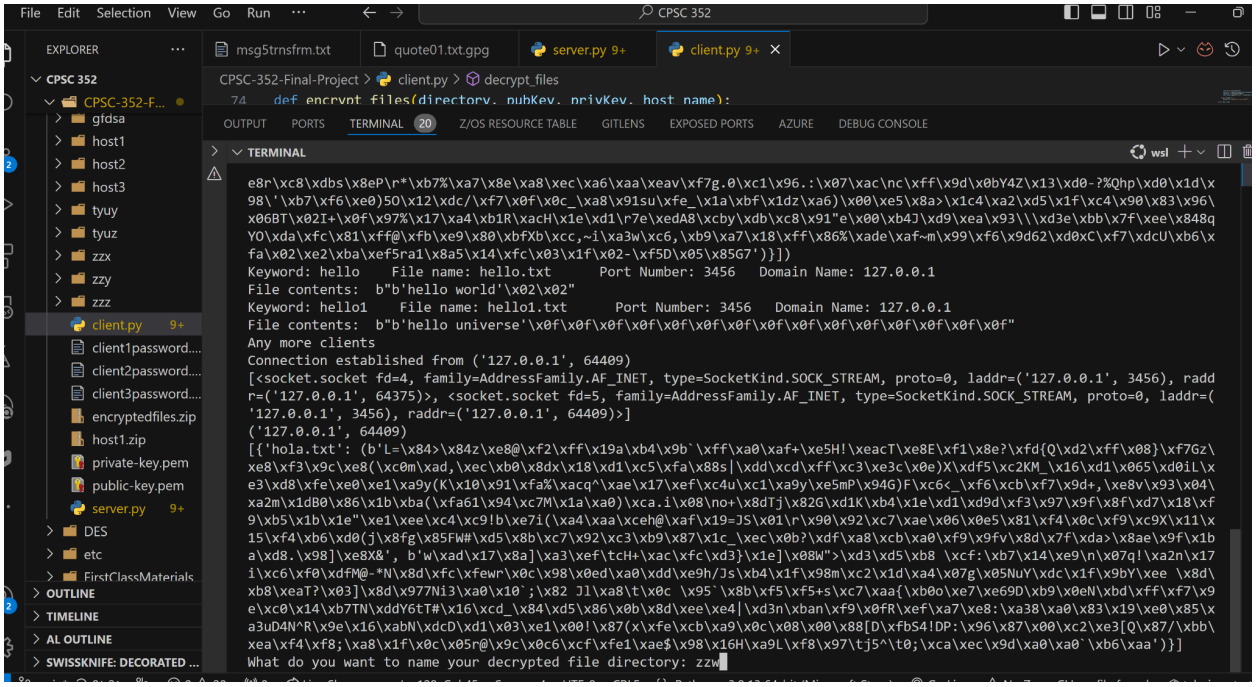

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > client.py > decrypt_files
def encrypt_files(directory, pubKey, privKey, host_name):
OUTPUT PORTS TERMINAL 20 Z/OS RESOURCE TABLE GITLENS EXPOSED PORTS AZURE DEBUG CONSOLE
TERMINAL
kushpatelj86@DESKTOP-T9949G2:/mnt/c/Users/Kush/myworkspace/CPSC 352/CPSC-352-Final-Project$ python3 server.py 3456 127.0.0.1
Server is listening on 127.0.0.1:3456
Connection established from ('127.0.0.1', 64287)
[<socket.socket fd=4, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 3456), raddr=('127.0.0.1', 64287)>]
('127.0.0.1', 64287)
[{'hello.txt': (b'\x04N\xb92#p]\xbd"\xe5\x7f\x820\xcf\xeb9\xfc\t\xb6\xcd-4\xdd\xe0% \xd7\xa1_3\x90\xc4\xe2\x06>\x80\xc4\xa2\xf5\x81\x86c\x8d\xdd\xf0\xd5\x07\xae\xa5\x83\xc96V\x8f1\xa7rX\xfeh\x1e\x18\xd2X\x00b\xfdph\xc4\x83\x0eJ]\x88\xfa\xd0A(5\xce\xe2\xc1\x0et\x15\x9e_ \xd0\x06\x85\xe5;]F*!\xf5\xf1Kw-D\xdf5\x9du\x87g\xf1\xdd$\xfeC\x12\x9f[\xf6\x03i&9#\xd8\xbb\xa6U$c\xe4\x8b\xec>Z5\xca\xe3&-\xea1\x04\xee\xb451&\x8a9t\xff#4\x91\x010;\xebc\xb7G\xedZ\xbb\xc5$\xf2\x16W\xccZ\xc5\x81\x9e\x9b\'\x9f\x06{\xcbA\xb6\xa3M\xdf\xbb\xd5\xdei\x1b}\xbe\x1dK\xa9@\x04\xe7\xec\xb4C\x91\xcf\xeb^\x87\xce6"r\r\x8f\xad\xdeZ\xe9\xa1o\x12\xbaa\x86\x95v\x9d\xbd\xeb\xb1\xf2\xa5^M\xceg\x87o\xef\xcb\x83L\xfaT\xf0\x9c\xaa\xceFu|\xba\x1d\x13\n\xbb', b"6\x91\x95\xda\xe1\xda\x10\xee\x07\\\x87\x9b\xc4\xb9\xbe\xba\x04\xa3D\xf19\x11\xc8(w\xb4\x90\x14\xdd}V\xc3\xf3\x9d\xf2a\xc4}'\x84\x86\xdb\xb8\x1f\x03\xcd\xf9\xe64\x98\xbfp\xdf\x9b\xe4\xa4\xa1(=\xd9\x1a\xc6\xc7\x04\xef*6\xf2\xa0\xac+\xf7OPc\xbf\xf5+X!\xfcQ\x93ou\xc1\x0f\x81\xe9\xf1&\xd9\xa1\x9d>G\xafvUHYARg\xdb\xc3\xce\xf3\xa4\x85\xd34\xa4P\xe2:}\x85\xbc\xe4\xa6\xdfE\x90\xeb\x92\xe0+\x8d\xca\x84\xdc\xe9\xaa\xfd\xc2D\x906\xfc1\x7fw\xf1\xbde\xad)\x1e\x18I,A\xf5\x87\xf5\t\xeb\x90\xbc\x94\xfc\xb2\xed\xb1`\x9ao^?\xcbF\x03\x1fJ\xdc~35w\xe7\x90\x8f\xbd\x1f\xdfh\x06\xf5e\xe6\\\xa1\x8b\xef\xb6\x04\xd4\xa8j\x83y\xf0\xbf\x9a\xc4\x7f\x90\x14\x10\xfb\xe7F\x9c?>2\x14\xd4\x87\x91\x18\xcd-\x0c\xa9<\xbf\x0b\xb6\xb6\xd7|]\x10u\xd7R$F\r \x14\xdf\xde\xb5\xe9\x10\xa2\x99g\x07_R\xae\x14")}, {'hello1.txt': (b'\x9aj\xe93\xb9\'U\x8c}\x91j\x08~\xe2nl\xf4\x0f\x98\xf2\xc0\x95tU\xceQ\xb6\xd6\xe3\xf7\xf2G2[i\xf2QXy\x9f\xb2S:\xf6\x99!Q\xae\x8f(\x03\x0cyom1&\x9d\xc8\x9f\x81\x9f1ve\x99\x9c\xf8\xfd:\x84\x8f\x1bJ\x89\xf0\xd3\xbdP\xb43\x8c\x16\xcf3W\x1c\x7f\xc6\x1c\x80\x99/\x88\xe7I\x92\xb9C\xa6c\x85\xd2\x8a\x15\x0e4\x89\xd8\x0b\x9d\xa3c\x8cO\x13\x95Bo\xa2.(\xfa\x16\x1c\x87\xc6%\xefn\xd0\xe1\x07*\x02\xe0\xf38\xbfs\xea\xfc\xbc\xabs\x16;\x9f1\xea\x0c2,\xaa\x86\x96/\x16\x8b\xac\xf8\xec\xd0P4\x81n\xfd\xa3\x9ceLX\xf2\xc9\xd8\xb2\xf0D\x14p\x1dZq6Y\xb2)\xcb\x87\x97F\x90\x90\x07\xaf\x06\xfd\xd8g\xff\xe0\xf3\xb8\xad\xedh\x1b\x89\x7fx&\r\xbeW\xe1[\xf1\xfa\x8f\xb99\x1e\'}\xa04\xa5j#k\xdfT%\xc6\xafhI8\xf0\x0e\xae\xa2`\x1f[I.\xe0\x12\x04\xc2\xc2)"', b'}\xa5a6D\x93\x8f \x06\xff5\xdcHCR^<^\xb2/\x8fb\xe8\xd8\xfbm\xfeeT\xac\x88\x0e\x08\xae\xa6\x87\xe5c\x87=\xecC\x0c\x85[\xad5\x8f\x12\x03$]\xdbB\xf6\xc8\xe5\xedlV\xe2\xf4_\xaf\xcd\xfb\xcd\xbc\xec\x07\xfc\rq\xf9<@\x83\x04\xf5X\x1d\xa5\xfb\xdd\x1b\x18\x8e\x80\x86\xed8\xb6\xac\x85\xe6\xf3\x01^hS\x0cTM\xdbM\xc81\xb2V\xa0]M\x92\xc4\xd7\xbc\x99\xdd\x8fX\x18p\x88\xa9.\x03oe\x19\xe3\xb0\xdfT\x8fFu\xf4\xd7\xd3\xf1^\xae\xdd\xd5\xc9\xd3S&\xe7\xcb w\xd9\xbcw\x86.R\xc0\xca\xee\xa4L0R\x99M\xb9\x04\xfb<\x17ZSj]\xbc6\xa4\xa7R\xb70\xc2\xbb\xd6\x9c)\xe8\xaf{\x7f`U\xf1W\xf7\xa7\x98\x8c\xb47y\xd6\x8a\x95a\xfc\xf6\xf7\x04\x82\xa6\x06\xc1:\x8c8\x9d\xf3m\x87W\xb1/\x9d\xb4\x8b\xb4\x9c%x\x1e\xc3g>\xe7A^$Z\xb2[mj\xd5\xf0\xa2\xe0\xe3\xfd9\x83\x93\x87\x9d')}]
What do you want to name your decrypted file directory: Connection established from ('127.0.0.1', 64315)
[<socket.socket fd=4, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 3456), raddr=('127.0.0.1', 64287)>, <socket.socket fd=5, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 3456), raddr=('127.0.0.1', 64315)>]
('127.0.0.1', 64315)
[{'hola.txt': (b'$D;\xba\x7fN,\xc7\xf4\x0f\x12^\xcc>\xb8@\x0cp\x06\x8cz\xb2\xa1\x11\x03\xad_\x18\x9a\xf0\\\x80\xabg\x0bj\xd4\x99\xe5\xc9`w\xaf2\xbf\xe5\x10\xba\x08P\x80\xbaT\xb8\xa7.\xc3\xea\x9ff\x9b\x86rM|\xec\xe0\x17\xfc\x8dj9\x8c\xc7\x83\r\x9d\x10\x0fj\xcd|\xc8"\xc6\xd2\xb6\xa2.\r\xe0BZxV\xad\x17\xcbZ\xc9:3\x96\xef\xa1\x83\x86\x97\xa3\xd1\x81k1\x96\xd0\x0c7Nr\x1f\xa55\xe0 k\xf2\xc3\x11L\x84\xe8G\x7f\x14`\xdf\x98\xb1R\x105\xed\xa8\xb7Y\xb7\xa8*\x8cf\xdc\x86\xc7c\x04\x13N"~\x0f\x0f\xc04(\xd0\x8d\xbc\xa8\x1a\nsBH\x8d~\x11\x85\x80\xc9\xe8\xd2\xe5\x1a4\xbbA\xe1\t\xf1\x96 \x1d\xd3G\x0c\xc43M\n\xb3\xcb\x16$ZJ+y\xd6\xfdc\x99\xbd{+7X\xb1\xf7Y?\xa8g\x08z\x12.\'\x17\xa0s\xff\xc9\x97\xe2\xa7\x07\x9d\xcf\xd0\xf7\xe4/\xf4\xc7p\x96lg<\xbfN\xe3\x86\xa4\xdf\xef', b'3M]b0\x1eu\xb9\xf5\x84fks\xa6Y\xc7\xbe\x10\xf7@\x96\xb6\xd6T!\xa2;\xbf\xf1\xcb\xaf\xdc\x9d\xacM/x%-\xd8?B\xfe`\x08\xe0\x00\xb7\x06\xc3n6\xa9\xba@w\xfa95R\xc1\x98\xda4\\\xe5\xf7(\xdb\xd9\xd1Q\xb5\x82\xdf(q\xf1\x0c\xa7J\xddW\xc1w\x8478(0\xfa\xfa}Em\x89I\xb6#b\x94\xc8Ja\x97\x11\x15^ \x8a\xd0\xd0\xac\xd7\xa3\x90\x87\xb3\xa7\xfe\xba\xb1P.\x7f\x11\x9b=\xf7\xdf\xbe\x90\xbf\x8c\xa5\x12\x8d\x8d\xb3\xdaM\xf0]\xb9\xf6\xc8\x14\x0eW\x17\xbaiX\xf7w^jm\x93\x89\xed\x1c\x10;\x90\xff\x17\x050O\x8a\xbc\xeej[+\x94\x95\xe8 \x1d\x95;\xbb\xf9{\x95VA\x90\xef\x14\x0274IhG\x03!+\xf4B\rr\xeb\xe5#`"\x99Km\x9b*\x86\x10\xac\x1d\xa5\xb4\x9c\xfb\x01\x1c\x15\x12\x05\xf2b*j\xc2\xa2#\x1b^\r\x93I\xe0yz\x19\xcew\xa3\x9d\xd9[_X\xe8\xac\xd0o\xaf')}]

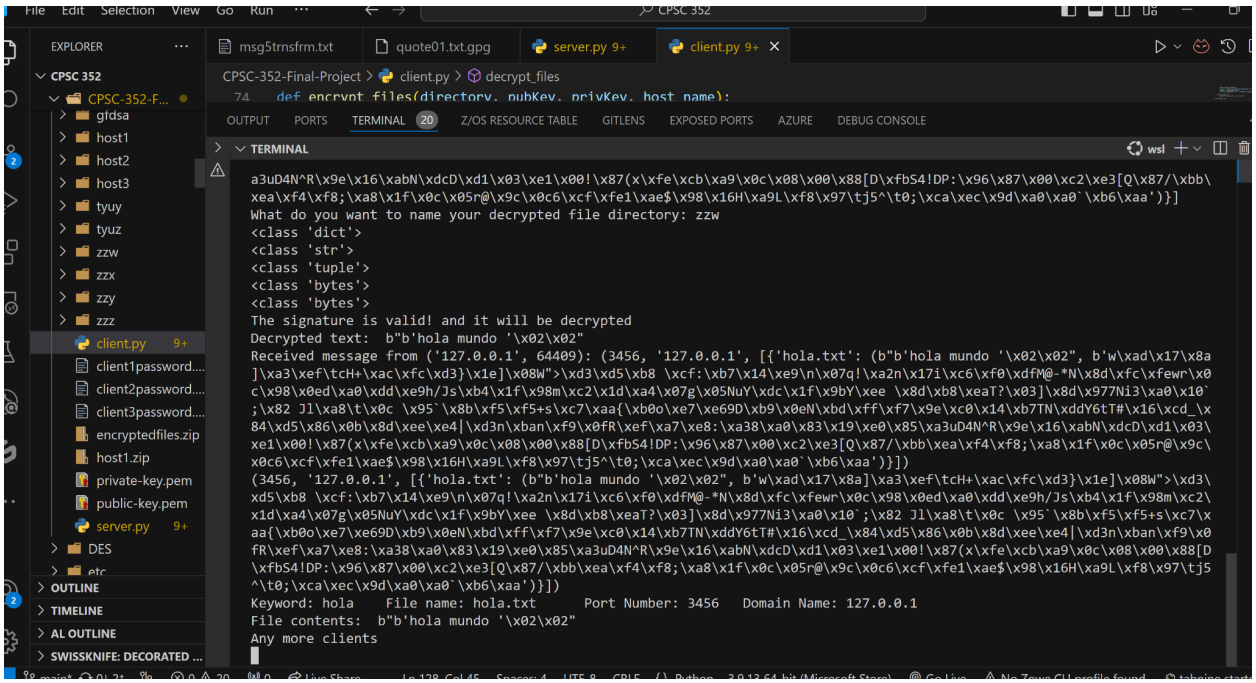

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > client.py > decrypt_files
OUTPUT PORTS TERMINAL 20 Z/OS RESOURCE TABLE GITLENS EXPOSED PORTS AZURE DEBUG CONSOLE
TERMINAL
wsl
kushpatelj86@DESKTOP-T9949G2:/mnt/c/Users/Kush/myworkspace/CPSC 352/CPSC-352-Final-Project$ python3 server.py 3456 127.0.0.1
Server is listening on 127.0.0.1:3456
CPSC-352-F...
gfdsa
host1
host2
host3
tyuy
tyuz
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
host1.zip
private-key.pem
public-key.pem
server.py 9+
DES
etc
FirstClassMaterials
fixfiles
client.py
server.py
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...

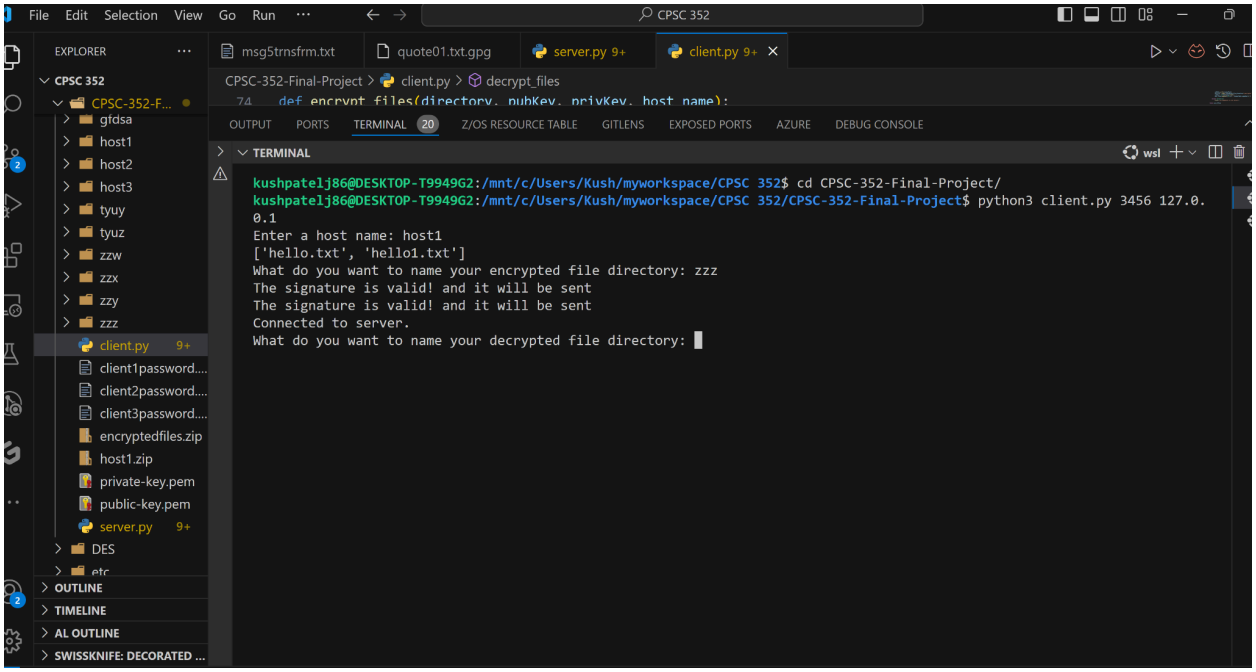

OUTPUT PORTS TERMINAL 20 Z/OS RESOURCE TABLE GITLENS EXPOSED PORTS AZURE DEBUG CONSOLE
TERMINAL
wsl
kushpatelj86@DESKTOP-T9949G2:/mnt/c/Users/Kush/myworkspace/CPSC 352/CPSC-352-Final-Project$ python3 server.py 3456 127.0.0.1
Server is listening on 127.0.0.1:3456
Connection established from ('127.0.0.1', 64375)
[<socket.socket fd=4, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 3456), raddr=('127.0.0.1', 64375)>]
('127.0.0.1', 64375)
[{'hello.txt': (b'\x0e\x04K\x03\x7fB&\x90\xf3\x00\x9a\t\xd3\xa408\xd2oR\xbd\x9e\x7f>\xf0&N\x8c5\xc3\xbdR\xa8\x1d*@\xdd\x1e\x92\x9ak\xc8\xe6=\xae`\xbc\xa85\xbaj\xbaX\x89\x92?<\xff[\xe0b\xd8i\xd7\xb5l\rbb-l&\x02\x07\x16\xec\xcc\x1a!\xf5JP\xca\xe5F\x9c\xf1\x00o\xed\xf2\xd9\xff\xd1\x0b\xc6m\xb4\xf8\xc0\xfe\xbc\xd0k6(\xbec\xe3v\x9a\xce\x81I\x80=\x85\x94\x88\xa7\xe2\xaa\x0f8\x88Q\x97y\x0c\x0f\xd7\xc1\x9f\x94\xbc\x8a\xd1\xd5|\x0f6\xe8\xefA\xa5\x8d\x871\x9c\x07.\x05ukw^\xf2\x9e\x81\x89\xdaN.\xa3q\xad\x929\x88\xf2+\xa3\xb1\xbe\xfc>\x06=r]\xcc\xe7{\xa6\xc6\xc4\xb4{\x7f\xc7\xe0\x15\xf8\xa4\xba]\xa6\xcd\xcf\xb4T`b8 \xedM1Z\xca)\t\x19;\x8e\x13\xec\x9d=A\xd0z\x85\xb6\x92(\x1d\xe9p\xb4\xa9\xc6\x86\xc4\xccwJJ#R\xc5\x19\xdc\x85\xe0\r\xa7\xa5\x1d\x93)\xd4\xeaz\x1a1\x98', b'A\xbdH<\xadMR\xc8\xa7\x19\xa9\xa1\x99\xbeB\\\n\x06f,\x97I\xe5\x88\x8e\xa6\xa7\xa6s\n(^\x10Y\xa2e\x1f\x86\xaf\xd7\xb4S<\xc6Na\xfc\xf9KO\xdeFb\x0e\x83C\x1eQ\x83\xc88\x1c\x98b\x1a\x1dQ\x075\x89\xcb7L\xdd\x9a\x1d\x99\xbf\xa0\xc0\xadG\xdb\x02\xebp\xb3\x83\x99R\x88\xcc.n\x11\xea\x91\xdc\xfe\xd7\xb3\x1e#\xa3\xb6\x14\xfd\xe8s\xd1O;\x03d@T\xc9\xc9\xbf\xeb\xa86T\x8e\xfeC\xa6%%2\x9c\xb4\xf9H\xd2\xb6U\r\xae(\xeb\x0f}\\\xe2\x93\x05\x19\xeb\xbb\xe1\xc6\xf9\x1a\xf1#\x12\x97K\xc0\xe2\x00\xfct\x04b\xb1y\xea\t%\x17\xa4\x84\xf6\xd5\xa7\x8d\xdc\xa8\xa05\xb8\xb3Y\x87\xe0)\xef\xcd6\xb4\x88\x16?\x87\xe9Cjq\xd5\xdb\xf5\xc7l\xcd\x1a\xeb(\xd7\xebX\xcb\x01\xab\xe4/\xb3\xadh\x06z`F\\\x87ld\x18]A\xe5\x9f\x9f\xac\x18\xce\x17pi_\xd5\xa8\xd8>\xf7\xc9h\xc1V\x80\x1aAB\xe8\xe9')}, {'hello1.txt': (b"Q'\xa7\xca\xf6 ]\xf9`\x90OD\x87.\xcbR`\xca\xb7\xab.\xc6\x19e\xf6\xd2\x9a\xf1\x04|\xa2\xa9\xbe\x7f\x87L\xc8\\\xd7F |>\xd3\xf0\x81\xaf\x0b\x15^\xb9\x93\x19\xf1\x82\xed\xc7\xfe\x9d\xde\xb6\xb7\xf9\n#\x90\xccr\xea\xcc\x0b\x89\x84p\x98\xcc\x81cE\xe1\x07\xde\xc8\xc4\xf9l2:S6?\x87\xe4\x02_!\x99#\x8e;\xebF\x11\xd8%\x86\xc22E*HrYx\xc7F\x12\xc2\xad\xcd\x87{f\x9f1\xa8^\x1e1\xeb\xdc\xdb\x93#O\x16\xfd'=&\xb9e\xcd\x92\x16\x17@\xe9\xd5\xc9\x98?\xeb#\x07\xf0Y@\x83\xaaY\xc0:\x82\xb3\x85h\x9c\x90\xe55\xebLa\xfc5U\xb2\xfcJ6pL\x9a\x18\x9eG\xd8\xbdL}z\xbc\xa0Ro\xc0\x9aph\xc5IT.\x7f\xc1CqE\xf9\xe3\xbd\xfd\x1e\x91\x05\xeb\x08\x90cO Q\xa4\xfb\x8d\xbe=\x9e\xec\xc9x\xf8s\t{\x1f\xec\xf2\xd8\xee\xb1\nn\xca\xb7d\x0f\x19,%m\xe59H(", b'?\x16\xdf\xd1\xe2\xbe5\x84\xb7\xee\xa8\xcf\x95\x0c\xfe\xb9R\xb8\x98e"\xaa\xb2\x04m/E\xd5\xa6v\xb3\xd8T=-6\xe7\xab\xad\xbdN\x94h[\xd4\xe1\x9b\x85G\xddtnq\x08,\xd2\xd1\xe8r\xc8\xdbs\x8eP\r*\xb7%\xa7\x8e\xa8\xec\xa6\xaa\xeav\xf7g.0\xc1\x96.:\x07\xac\nc\xff\x9d\x0bY4Z\x13\xd0-?%Qhp\xd0\x1

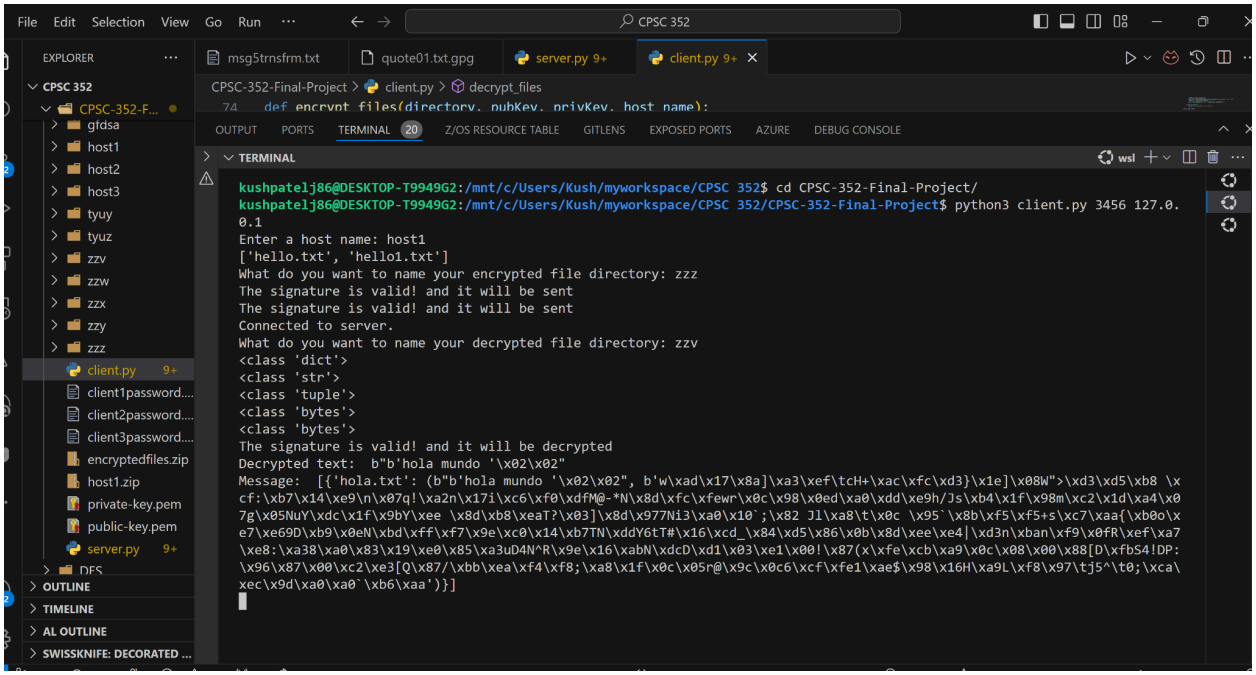

TERMINAL
wsl
r=('127.0.0.1', 64375)>]
('127.0.0.1', 64375)
[{'hello.txt': (b'\x0e\x04K\x03\x7fB&\x90\xf3\x00\x9a\t\xd3\xa408\xd2oR\xbd\x9e\x7f>\xf0&N\x8c5\xc3\xbdR\xa8\x1d*@\xdd\x1e\x92\x9ak\xc8\xe6=\xae`\xbc\xa85\xbaj\xbaX\x89\x92?<\xff[\xe0b\xd8i\xd7\xb5l\rbb-l&\x02\x07\x16\xec\xcc\x1a!\xf5JP\xca\xe5F\x9c\xf1\x00o\xed\xf2\xd9\xff\xd1\x0b\xc6m\xb4\xf8\xc0\xfe\xbc\xd0k6(\xbec\xe3v\x9a\xce\x81I\x80=\x85\x94\x88\xa7\xe2\xaa\x0f8\x88Q\x97y\x0c\x0f\xd7\xc1\x9f\x94\xbc\x8a\xd1\xd5|\x0f6\xe8\xefA\xa5\x8d\x871\x9c\x07.\x05ukw^\xf2\x9e\x81\x89\xdaN.\xa3q\xad\x929\x88\xf2+\xa3\xb1\xbe\xfc>\x06=r]\xcc\xe7{\xa6\xc6\xc4\xb4{\x7f\xc7\xe0\x15\xf8\xa4\xba]\xa6\xcd\xcf\xb4T`b8 \xedM1Z\xca)\t\x19;\x8e\x13\xec\x9d=A\xd0z\x85\xb6\x92(\x1d\xe9p\xb4\xa9\xc6\x86\xc4\xccwJJ#R\xc5\x19\xdc\x85\xe0\r\xa7\xa5\x1d\x93)\xd4\xeaz\x1a1\x98', b'A\xbdH<\xadMR\xc8\xa7\x19\xa9\xa1\x99\xbeB\\\n\x06f,\x97I\xe5\x88\x8e\xa6\xa7\xa6s\n(^\x10Y\xa2e\x1f\x86\xaf\xd7\xb4S<\xc6Na\xfc\xf9KO\xdeFb\x0e\x83C\x1eQ\x83\xc88\x1c\x98b\x1a\x1dQ\x075\x89\xcb7L\xdd\x9a\x1d\x99\xbf\xa0\xc0\xadG\xdb\x02\xebp\xb3\x83\x99R\x88\xcc.n\x11\xea\x91\xdc\xfe\xd7\xb3\x1e#\xa3\xb6\x14\xfd\xe8s\xd1O;\x03d@T\xc9\xc9\xbf\xeb\xa86T\x8e\xfeC\xa6%%2\x9c\xb4\xf9H\xd2\xb6U\r\xae(\xeb\x0f}\\\xe2\x93\x05\x19\xeb\xbb\xe1\xc6\xf9\x1a\xf1#\x12\x97K\xc0\xe2\x00\xfct\x04b\xb1y\xea\t%\x17\xa4\x84\xf6\xd5\xa7\x8d\xdc\xa8\xa05\xb8\xb3Y\x87\xe0)\xef\xcd6\xb4\x88\x16?\x87\xe9Cjq\xd5\xdb\xf5\xc7l\xcd\x1a\xeb(\xd7\xebX\xcb\x01\xab\xe4/\xb3\xadh\x06z`F\\\x87ld\x18]A\xe5\x9f\x9f\xac\x18\xce\x17pi_\xd5\xa8\xd8>\xf7\xc9h\xc1V\x80\x1aAB\xe8\xe9')}, {'hello1.txt': (b"Q'\xa7\xca\xf6 ]\xf9`\x90OD\x87.\xcbR`\xca\xb7\xab.\xc6\x19e\xf6\xd2\x9a\xf1\x04|\xa2\xa9\xbe\x7f\x87L\xc8\\\xd7F |>\xd3\xf0\x81\xaf\x0b\x15^\xb9\x93\x19\xf1\x82\xed\xc7\xfe\x9d\xde\xb6\xb7\xf9\n#\x90\xccr\xea\xcc\x0b\x89\x84p\x98\xcc\x81cE\xe1\x07\xde\xc8\xc4\xf9l2:S6?\x87\xe4\x02_!\x99#\x8e;\xebF\x11\xd8%\x86\xc22E*HrYx\xc7F\x12\xc2\xad\xcd\x87{f\x9f1\xa8^\x1el\xeb\xdc\xdb\x93#O\x16\xfd'=&\xb9e\xcd\x92\x16\x17@\xe9\xd5\xc9\x98?\xeb#\x07\xf0Y@\x83\xaaY\xc0:\x82\xb3\x85h\x9c\x90\xe55\xebLa\xfc5U\xb2\xfcJ6pL\x9a\x18\x9eG\xd8\xbdL}z\xbc\xa0Ro\xc0\x9aph\xc5IT.\x7f\xc1CqE\xf9\xe3\xbd\xfd\x1e\x91\x05\xeb\x08\x90cO Q\xa4\xfb\x8d\xbe=\x9e\xec\xc9x\xf8s\t{\x1f\xec\xf2\xd8\xee\xb1\nn\xca\xb7d\x0f\x19,%m\xe59H(", b'?\x16\xdf\xd1\xe2\xbe5\x84\xb7\xee\xa8\xcf\x95\x0c\xfe\xb9R\xb8\x98e"\xaa\xb2\x04m/E\xd5\xa6v\xb3\xd8T=-6\xe7\xab\xad\xbdN\x94h[\xd4\xe1\x9b\x85G\xddtnq\x08,\xd2\xd1\xe8r\xc8\xdbs\x8eP\r*\xb7%\xa7\x8e\xa8\xec\xa6\xaa\xeav\xf7g.0\xc1\x96.:\x07\xac\nc\xff\x9d\x0bY4Z\x13\xd0-?%Qhp\xd0\x1d\x98\'\xb7\xf6\xe0)5O\x12\xdc/\xf7\x0f\x0c_\xa8\x91su\xfe_\x1a\xbf\x1dz\xa6)\x00\xe5\x8a>\x1c4\xa2\xd5\x1f\xc4\x90\x83\x96\x06BT\x02I+\x0f\x97%\x17\xa4\xb1R\xacH\x1e\xd1\r7e\xedA8\xcby\xdb\xc8\x91"e\x00\xb4J\xd9\xea\x93\\\xd3e\xbb\x7f\xee\x848qYO\xda\xfc\x81\xff@\xfb\xe9\x80\xbfXb\xcc,~i\xa3w\xc6,\xb9\xa7\x18\xff\x86%\xade\xaf~m\x99\xf6\x9d62\xd0xC\xf7\xdcU\xb6\xfa\x02\xe2\xba\xef5ra1\x8a5\x14\xfc\x03\x1f\x02-\xf5D\x05\x85G7')}]
What do you want to name your decrypted file directory:

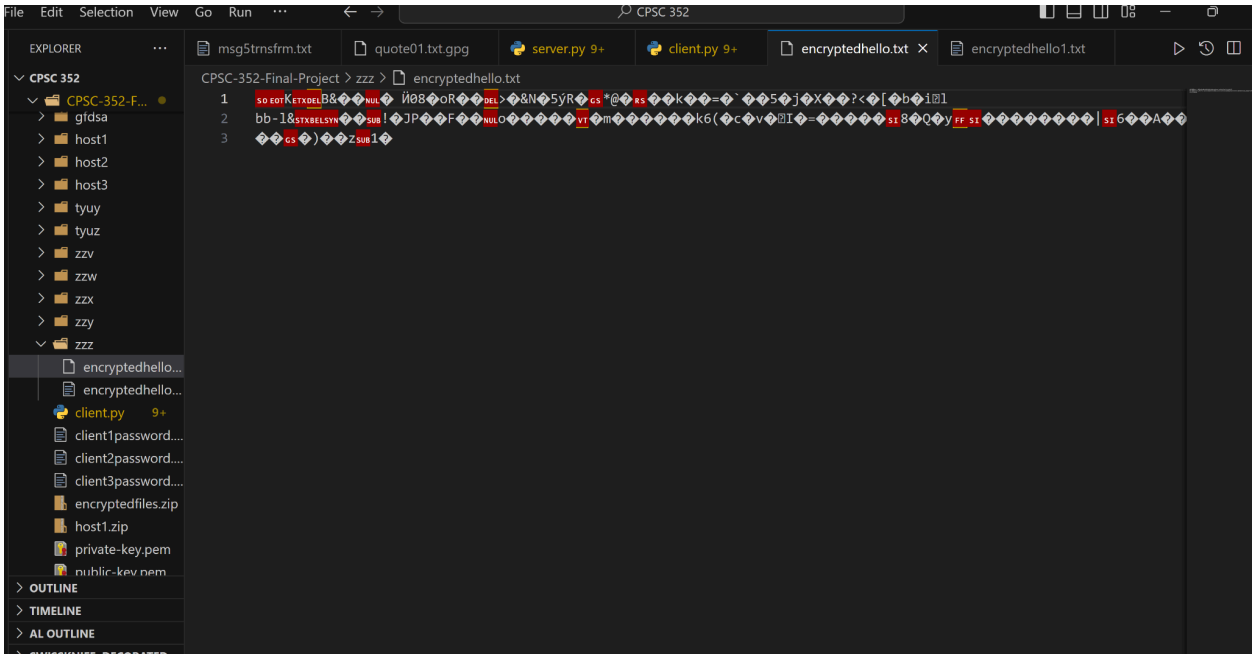

EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > client.py > decrypt_files
def encrypt_files(directory, pubKey, privKey, host_name):
OUTPUT
PORTS
TERMINAL 20
Z/OS RESOURCE TABLE
GITLENS
EXPOSED PORTS
AZURE
DEBUG CONSOLE
TERMINAL
wsl
CPSC 352
CPSC-352-F...
gfdsa
host1
host2
host3
tyuy
tyuz
zzy
zzz
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
host1.zip
private-key.pem
public-key.pem
server.py 9+
DES
etc
FirstClassMaterials
fixfiles
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...
Keyword: hello File name: hello.txt Port Number: 3456 Domain Name: 127.0.0.1
File contents: b"b'hello world'\x02\x02"
Keyword: hello1 File name: hello1.txt Port Number: 3456 Domain Name: 127.0.0.1
File contents: b"b'hello universe'\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f"
Any more clients

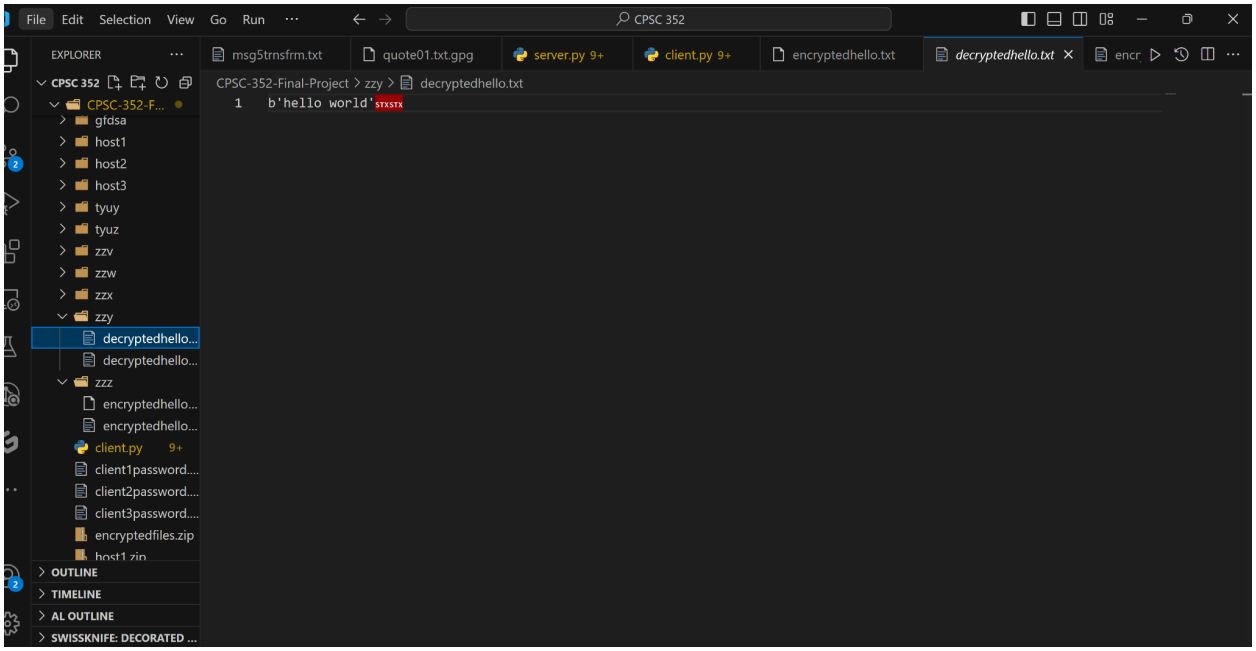

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > client.py > decrypt_files
74 def encrypt_files(directory, pubKey, privKey, host_name):
OUTPUT PORTS TERMINAL 20 Z/OS RESOURCE TABLE GITLENS EXPOSED PORTS AZURE DEBUG CONSOLE
TERMINAL
wsl
e8r\xc8\xdbs\x8eP\r*\xb7%\xa7\x8e\xa8\xec\xa6\xaa\xeav\xf7g.0\xc1\x96.:\x07\xac\nc\xff\x9d\x0bY4Z\x13\xd0-?%Qhp\xd0\x1d\x
98\'\xb7\xf6\xe0)5O\x12\xdc/\xf7\x0f\x0c_\xa8\x91su\xfe_\x1a\xbf\x1dz\xa6)\x00\xe5\x8a>\x1c4\xa2\xd5\x1f\xc4\x90\x83\x96\
x06BT\x02I+\x0f\x97%\x17\xa4\xb1R\xacH\x1e\xd1\r7e\xedA8\xcby\xdb\xc8\x91"e\x00\xb4J\xd9\xea\x93\\\xd3e\xbb\x7f\xee\x848q
YO\xda\xfc\x81\xff@\xfb\xe9\x80\xbfXb\xcc,~i\xa3w\xc6,\xb9\xa7\x18\xff\x86%\xade\xaf~m\x99\xf6\x9d62\xd0xC\xf7\xdcU\xb6\x
fa\x02\xe2\xba\xef5ra1\x8a5\x14\xfc\x03\x1f\x02-\xf5D\x05\x85G7')}]])
Keyword: hello File name: hello.txt Port Number: 3456 Domain Name: 127.0.0.1
File contents: b"b'hello world'\x02\x02"
Keyword: hello1 File name: hello1.txt Port Number: 3456 Domain Name: 127.0.0.1
File contents: b"b'hello universe'\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f"
Any more clients
Connection established from ('127.0.0.1', 64409)
[<socket.socket fd=4, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 3456), radd
r=('127.0.0.1', 64375)>, <socket.socket fd=5, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=(
'127.0.0.1', 3456), raddr=('127.0.0.1', 64409)>]
('127.0.0.1', 64409)
[{'hola.txt': (b'L=\x84>\x84z\xe8@\xf2\xff\x19a\xb4\x9b`\xff\xa0\xaf+\xe5H!\xeacT\xe8E\xf1\x8e?\xfd{Q\xd2\xff\x08}\xf7Gz\
xe8\xf3\x9c\xe8(\xc0m\xad,\xec\xb0\x8dx\x18\xd1\xc5\xfa\x88s|\xdd\xcd\xff\xc3\xe3c\x0e)X\xdf5\xc2KM_\x16\xd1\x065\xd0iL\x
e3\xd8\xfe\xe0\xe1\xa9y(K\x10\x91\xfa%\xacq^\xae\x17\xef\xc4u\xc1\xa9y\xe5mP\x94G)F\xc6<_\xf6\xcb\xf7\x9d+,\xe8v\x93\x04\
xa2m\x1dB0\x86\x1b\xba(\xfa61\x94\xc7M\x1a\xa0)\xca.i\x08\no+\x8dTj\x82G\xd1K\xb4\x1e\xd1\xd9d\xf3\x97\x9f\x8f\xd7\x18\xf
9\xb5\x1b\x1e"\xe1\xee\xc4\xc9!b\xe7i(\xa4\xaa\xceh@\xaf\x19=JS\x01\r\x90\x92\xc7\xae\x06\x0e5\x81\xf4\x0c\xf9\xc9X\x11\x
15\xf4\xb6\xd0(j\x8fg\x85FW#\xd5\x8b\xc7\x92\xc3\xb9\x87\x1c_\xec\x0b?\xdf\xa8\xcb\xa0\xf9\x9fv\x8d\x7f\xda>\x8ae\x9f\x1b
a\xd8.\x98]\xe8X&', b'w\xad\x17\x8a]\xa3\xef\tcH+\xac\xfc\xd3}\x1e]\x08W">\xd3\xd5\xb8 \xcf:\xb7\x14\xe9\n\x07q!\xa2n\x17
i\xc6\xf0\xdfM@-*N\x8d\xfc\xfewr\x0c\x98\x0ed\xa0\xdd\xe9h/Js\xb4\x1f\x98m\xc2\x1d\xa4\x07g\x05NuY\xdc\x1f\x9bY\xee \x8d\
xb8\xeaT?\x03]\x8d\x977Ni3\xa0\x10`;\x82 Jl\xa8\t\x0c \x95`\x8b\xf5\xf5+s\xc7\xaa{\xb0o\xe7\xe69D\xb9\x0eN\xbd\xff\xf7\x9
e\xc0\x14\xb7TN\xddY6tT#\x16\xcd_\x84\xd5\x86\x0b\x8d\xee\xe4|\xd3n\xban\xf9\x0fR\xef\xa7\xe8:\xa38\xa0\x83\x19\xe0\x85\x
a3uD4N^R\x9e\x16\xabN\xdcD\xd1\x03\xe1\x00!\x87(x\xfe\xcb\xa9\x0c\x08\x00\x88[D\xfbS4!DP:\x96\x87\x00\xc2\xe3[Q\x87/\xbb\
xea\xf4\xf8;\xa8\x1f\x0c\x05r@\x9c\x0c6\xcf\xfe1\xae$\x98\x16H\xa9L\xf8\x97\tj5^\t0;\xca\xec\x9d\xa0\xa0`\xb6\xaa')}]
What do you want to name your decrypted file directory: zzw


File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > client.py > decrypt_files
74 def encrypt_files(directory, pubKey, privKey, host_name):
OUTPUT PORTS TERMINAL 20 Z/OS RESOURCE TABLE GITLENS EXPOSED PORTS AZURE DEBUG CONSOLE
TERMINAL
wsl
a3uD4N^R\x9e\x16\xabN\xdcD\xd1\x03\xe1\x00!\x87(x\xfe\xcb\xa9\x0c\x08\x00\x88[D\xfbS4!DP:\x96\x87\x00\xc2\xe3[Q\x87/\xbb\
xea\xf4\xf8;\xa8\x1f\x0c\x05r@\x9c\x0c6\xcf\xfe1\xae$\x98\x16H\xa9L\xf8\x97\tj5^\t0;\xca\xec\x9d\xa0\xa0`\xb6\xaa')}]
What do you want to name your decrypted file directory: zzw
<class 'dict'>
<class 'str'>
<class 'tuple'>
<class 'bytes'>
<class 'bytes'>
The signature is valid! and it will be decrypted
Decrypted text: b"b'hola mundo '\x02\x02"
Received message from ('127.0.0.1', 64409): (3456, '127.0.0.1', [{'hola.txt': (b"b'hola mundo '\x02\x02", b'w\xad\x17\x8a
]\xa3\xef\tcH+\xac\xfc\xd3}\x1e]\x08W">\xd3\xd5\xb8 \xcf:\xb7\x14\xe9\n\x07q!\xa2n\x17i\xc6\xf0\xdfM@-*N\x8d\xfc\xfewr\x0
c\x98\x0ed\xa0\xdd\xe9h/Js\xb4\x1f\x98m\xc2\x1d\xa4\x07g\x05NuY\xdc\x1f\x9bY\xee \x8d\xb8\xeaT?\x03]\x8d\x977Ni3\xa0\x10`
;\x82 Jl\xa8\t\x0c \x95`\x8b\xf5\xf5+s\xc7\xaa{\xb0o\xe7\xe69D\xb9\x0eN\xbd\xff\xf7\x9e\xc0\x14\xb7TN\xddY6tT#\x16\xcd_\x
84\xd5\x86\x0b\x8d\xee\xe4|\xd3n\xban\xf9\x0fR\xef\xa7\xe8:\xa38\xa0\x83\x19\xe0\x85\xa3uD4N^R\x9e\x16\xabN\xdcD\xd1\x03\
xe1\x00!\x87(x\xfe\xcb\xa9\x0c\x08\x00\x88[D\xfbS4!DP:\x96\x87\x00\xc2\xe3[Q\x87/\xbb\xea\xf4\xf8;\xa8\x1f\x0c\x05r@\x9c\
x0c6\xcf\xfe1\xae$\x98\x16H\xa9L\xf8\x97\tj5^\t0;\xca\xec\x9d\xa0\xa0`\xb6\xaa')}]
(3456, '127.0.0.1', [{'hola.txt': (b"b'hola mundo '\x02\x02", b'w\xad\x17\x8a]\xa3\xef\tcH+\xac\xfc\xd3}\x1e]\x08W">\xd3\
xd5\xb8 \xcf:\xb7\x14\xe9\n\x07q!\xa2n\x17i\xc6\xf0\xdfM@-*N\x8d\xfc\xfewr\x0c\x98\x0ed\xa0\xdd\xe9h/Js\xb4\x1f\x98m\xc2\
x1d\xa4\x07g\x05NuY\xdc\x1f\x9bY\xee \x8d\xb8\xeaT?\x03]\x8d\x977Ni3\xa0\x10`;\x82 Jl\xa8\t\x0c \x95`\x8b\xf5\xf5+s\xc7\x
aa{\xb0o\xe7\xe69D\xb9\x0eN\xbd\xff\xf7\x9e\xc0\x14\xb7TN\xddY6tT#\x16\xcd_\x84\xd5\x86\x0b\x8d\xee\xe4|\xd3n\xban\xf9\x0
fR\xef\xa7\xe8:\xa38\xa0\x83\x19\xe0\x85\xa3uD4N^R\x9e\x16\xabN\xdcD\xd1\x03\xe1\x00!\x87(x\xfe\xcb\xa9\x0c\x08\x00\x88[D
\xfbS4!DP:\x96\x87\x00\xc2\xe3[Q\x87/\xbb\xea\xf4\xf8;\xa8\x1f\x0c\x05r@\x9c\x0c6\xcf\xfe1\xae$\x98\x16H\xa9L\xf8\x97\tj5
^\t0;\xca\xec\x9d\xa0\xa0`\xb6\xaa')}]])
Keyword: hola File name: hola.txt Port Number: 3456 Domain Name: 127.0.0.1
File contents: b"b'hola mundo '\x02\x02"
Any more clients

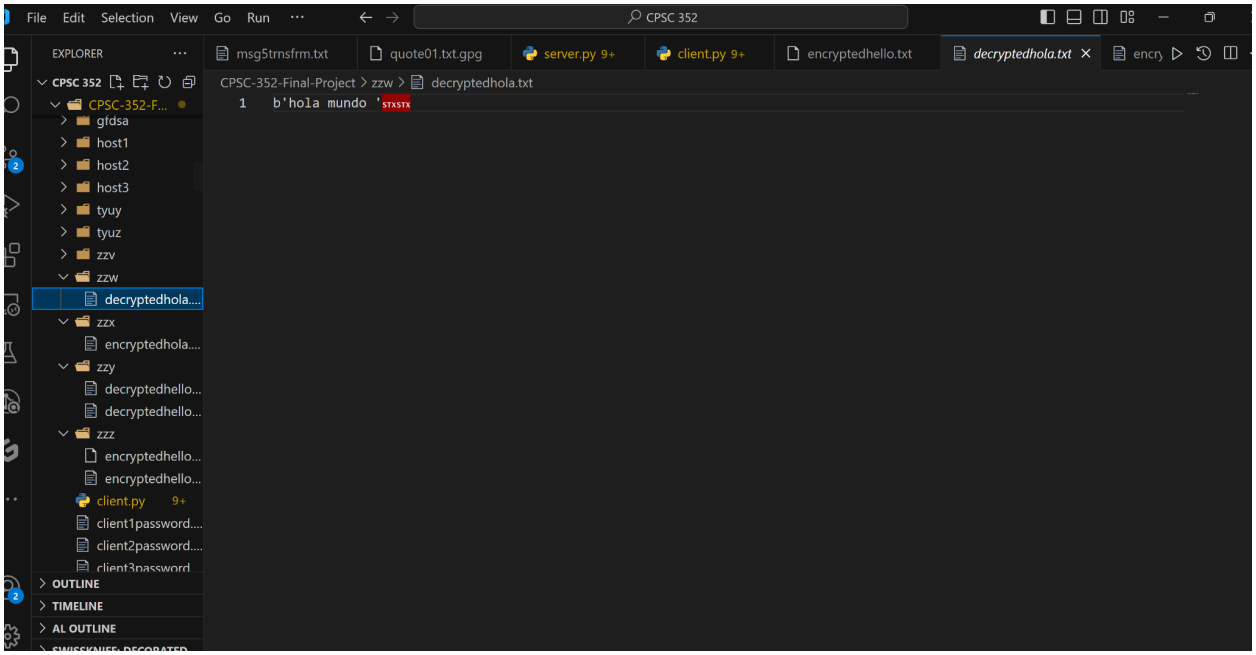

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > client.py > decrypt_files
def encrypt_files(directory, pubKey, privKey, host_name):
OUTPUT PORTS TERMINAL 20 Z/OS RESOURCE TABLE GITLENS EXPOSED PORTS AZURE DEBUG CONSOLE
TERMINAL
wsl
kushpatelj86@DESKTOP-T9949G2:/mnt/c/Users/Kush/myworkspace/CPSC 352$ cd CPSC-352-Final-Project/
kushpatelj86@DESKTOP-T9949G2:/mnt/c/Users/Kush/myworkspace/CPSC 352/CPSC-352-Final-Project$ python3 client.py 3456 127.0.0.1
Enter a host name: host1
['hello.txt', 'hello1.txt']
What do you want to name your encrypted file directory: zzz
The signature is valid! and it will be sent
The signature is valid! and it will be sent
Connected to server.
What do you want to name your decrypted file directory:
CPSC 352
gtdsa
host1
host2
host3
tyuy
tyuz
zzw
zzx
zzy
zzz
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
host1.zip
private-key.pem
public-key.pem
server.py 9+
DES
etc
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...

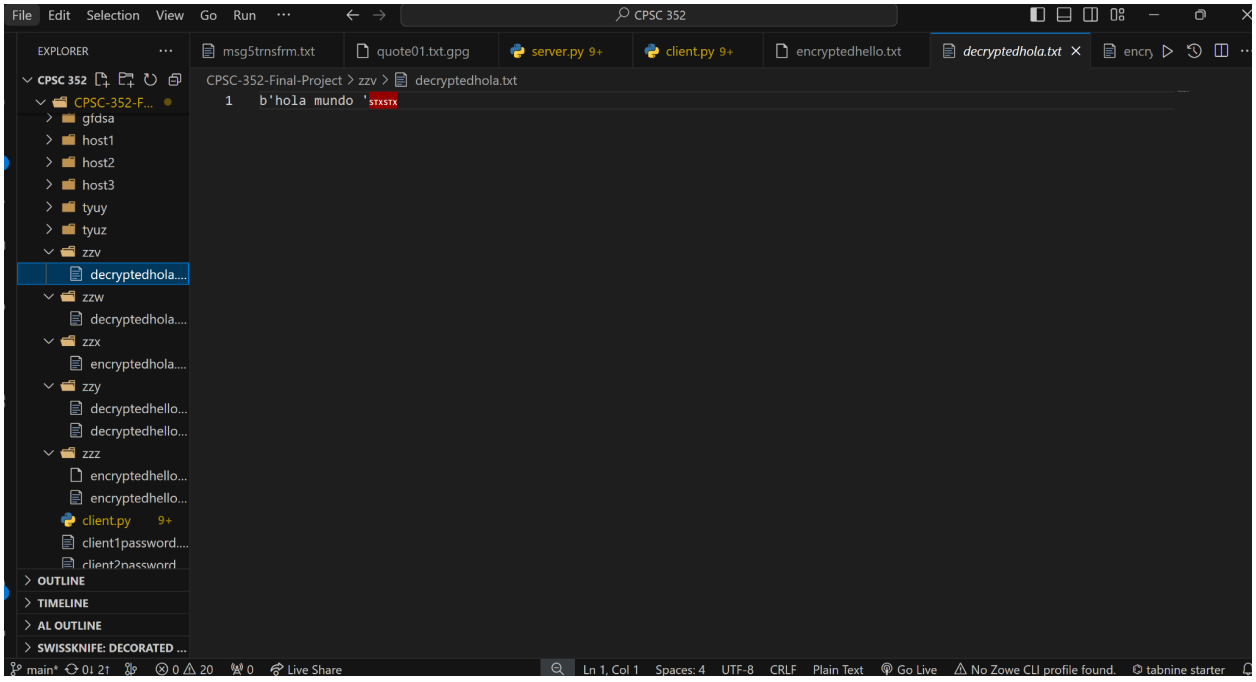

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > client.py > decrypt_files
def encrypt_files(directory, pubKey, privKey, host_name):
OUTPUT PORTS TERMINAL 20 Z/OS RESOURCE TABLE GITLENS EXPOSED PORTS AZURE DEBUG CONSOLE
TERMINAL
wsl
kushpatelj86@DESKTOP-T9949G2:/mnt/c/Users/Kush/myworkspace/CPSC 352$ cd CPSC-352-Final-Project/
kushpatelj86@DESKTOP-T9949G2:/mnt/c/Users/Kush/myworkspace/CPSC 352/CPSC-352-Final-Project$ python3 client.py 3456 127.0.0.1
Enter a host name: host1
['hello.txt', 'hello1.txt']
What do you want to name your encrypted file directory: zzz
The signature is valid! and it will be sent
The signature is valid! and it will be sent
Connected to server.
What do you want to name your decrypted file directory: zzv
<class 'dict'>
<class 'str'>
<class 'tuple'>
<class 'bytes'>
<class 'bytes'>
The signature is valid! and it will be decrypted
Decrypted text: b"b'hola mundo '\x02\x02"
Message: [{'hola.txt': (b"b'hola mundo '\x02\x02", b'w\xad\x17\x8a]\xa3\xef\tcH+\xac\xfc\xd3}\x1e]\x08W">\xd3\xd5\xb8 \xcf:\xb7\x14\xe9\n\x07q!\xa2n\x17i\xc6\xf0\xdfM@-*N\x8d\xfc\xfewr\x0c\x98\x0ed\xa0\xdd\xe9h/Js\xb4\x1f\x98m\xc2\x1d\xa4\x07g\x05NuY\xdc\x1f\x9bY\xee \x8d\xb8\xeaT?\x03]\x8d\x977Ni3\xa0\x10`;\x82 Jl\xa8\t\x0c \x95`\x8b\xf5\xf5+s\xc7\xaa{\xb0o\xe7\xe69D\xb9\x0eN\xbd\xff\xf7\x9e\xc0\x14\xb7TN\xddY6tT#\x16\xcd_\x84\xd5\x86\x0b\x8d\xee\xe4|\xd3n\xban\xf9\x0fR\xef\xa7\xe8:\xa38\xa0\x83\x19\xe0\x85\xa3uD4N^R\x9e\x16\xabN\xdcD\xd1\x03\xe1\x00!\x87(x\xfe\xcb\xa9\x0c\x08\x00\x88[D\xfbS4!DP:\x96\x87\x00\xc2\xe3[Q\x87/\xbb\xea\xf4\xf8;\xa8\x1f\x0c\x05r@\x9c\x0c6\xcf\xfe1\xae$\x98\x16H\xa9L\xf8\x97\tj5^\t0;\xca\xec\x9d\xa0\xa0`\xb6\xaa')}]
CPSC 352
gtdsa
host1
host2
host3
tyuy
tyuz
zzv
zzw
zzx
zzy
zzz
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
host1.zip
private-key.pem
public-key.pem
server.py 9+
DES
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...

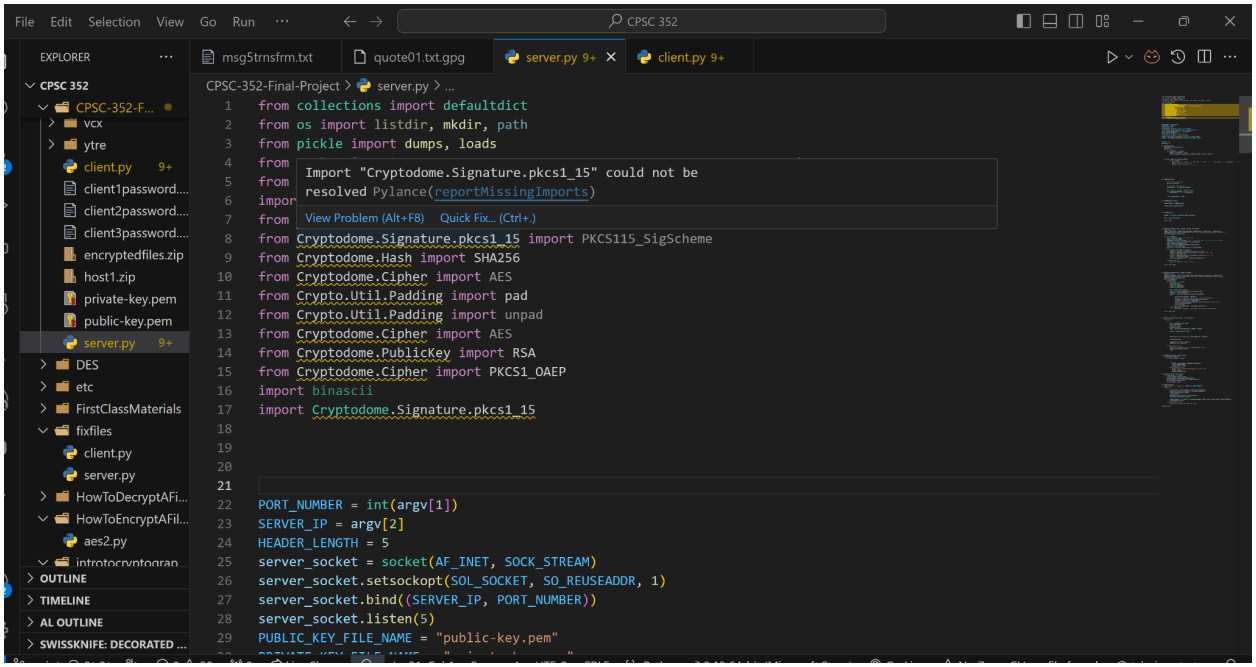

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
encryptedhello.txt
encryptedhello1.txt
CPSC-352-Final-Project > zzz > encryptedhello.txt
CPSC 352
CPSC-352-F...
gfdsa
host1
host2
host3
tyuy
tyuz
zzv
zzw
zzx
zzy
zzz
encryptedhello...
encryptedhello...
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
host1.zip
private-key.pem
public-key.pem
OUTLINE
TIMELINE
AL OUTLINE

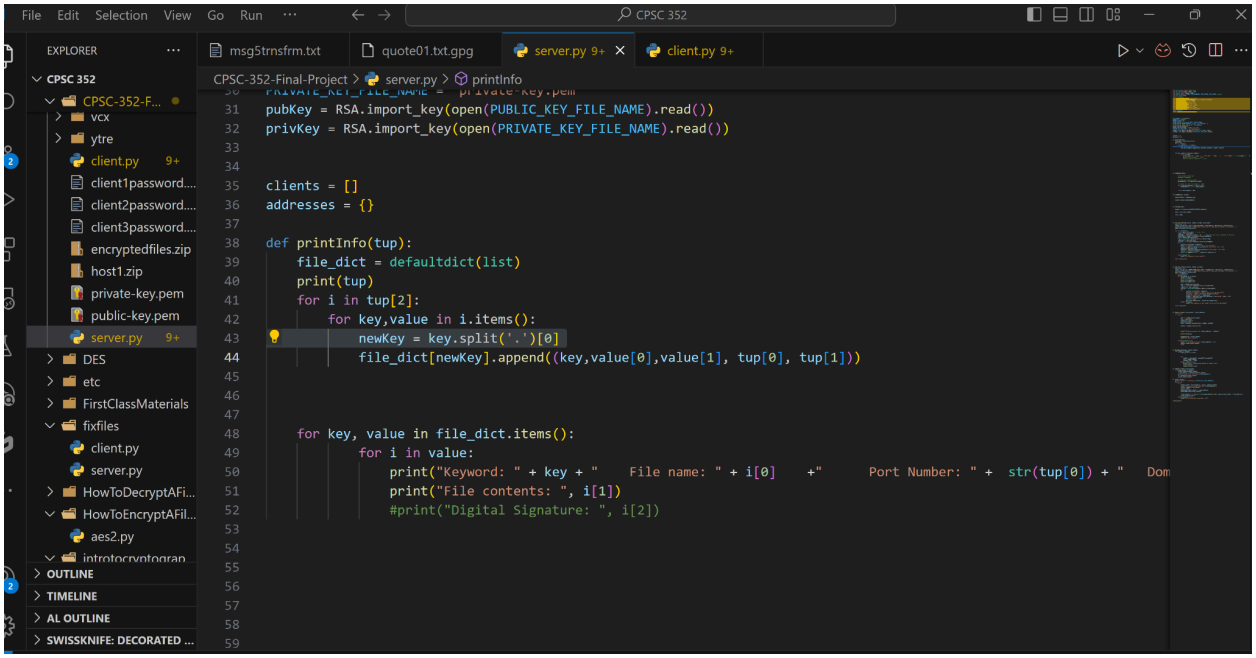

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
encryptedhello.txt
decryptedhello.txt
CPSC-352-Final-Project > zzy > decryptedhello.txt
b'hello world'
CPSC 352
CPSC-352-F...
gfdsa
host1
host2
host3
tyuy
tyuz
zzv
zzw
zzx
zzy
decryptedhello...
decryptedhello...
zzz
encryptedhello...
encryptedhello...
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...


File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
encryptedhello.txt
encryptedhola.txt
CPSC-352-Final-Project > zzx > encryptedhola.txt
CPSC 352
CPSC-352-F...
gfdsa
host1
host2
host3
tyuy
tyuz
zzv
zzw
zzx
encryptedhola....
zzy
decryptedhello...
decryptedhello...
zzz
encryptedhello...
encryptedhello...
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
OUTLINE
TIMELINE
AL OUTLINE

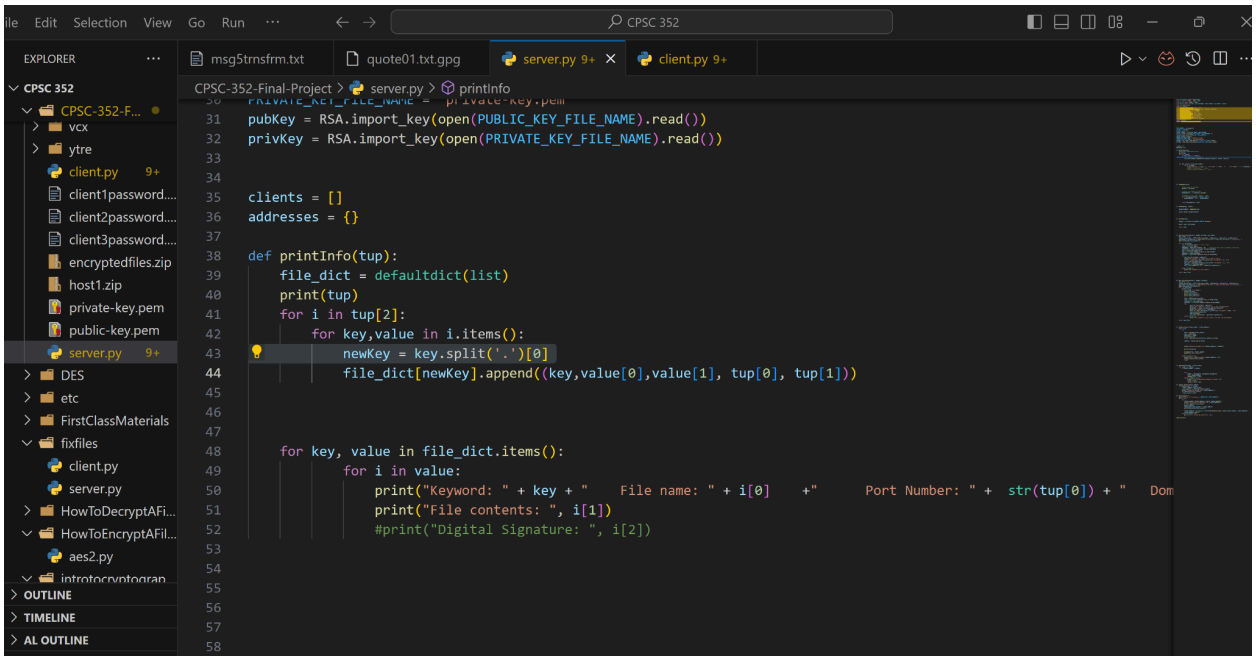

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
encryptedhello.txt
decryptedhola.txt
CPSC-352-Final-Project > zzw > decryptedhola.txt
1 b'hola mundo '
CPSC 352
CPSC-352-F...
gfdsa
host1
host2
host3
tyuy
tyuz
zzv
zzw
decryptedhola....
zzx
encryptedhola....
zzy
decryptedhello...
decryptedhello...
zzz
encryptedhello...
encryptedhello...
client.py 9+
client1password....
client2password....
OUTLINE
TIMELINE
AL OUTLINE


File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
encryptedhello.txt
decryptedhola.txt
CPSC-352-Final-Project > zzv > decryptedhola.txt
1 b'hola mundo 'STXSTX
CPSC 352
CPSC-352-F...
gfdsa
host1
host2
host3
tyuy
tyuz
zzv
decryptedhola....
zzw
decryptedhola....
zzx
encryptedhola....
zzy
decryptedhello...
decryptedhello...
zzz
encryptedhello...
encryptedhello...
client.py 9+
client1password....
client2password
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...
main* 0↓ 2↑ 0 20 0 Live Share
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Plain Text Go Live No Zowe CLI profile found. tabnine starter

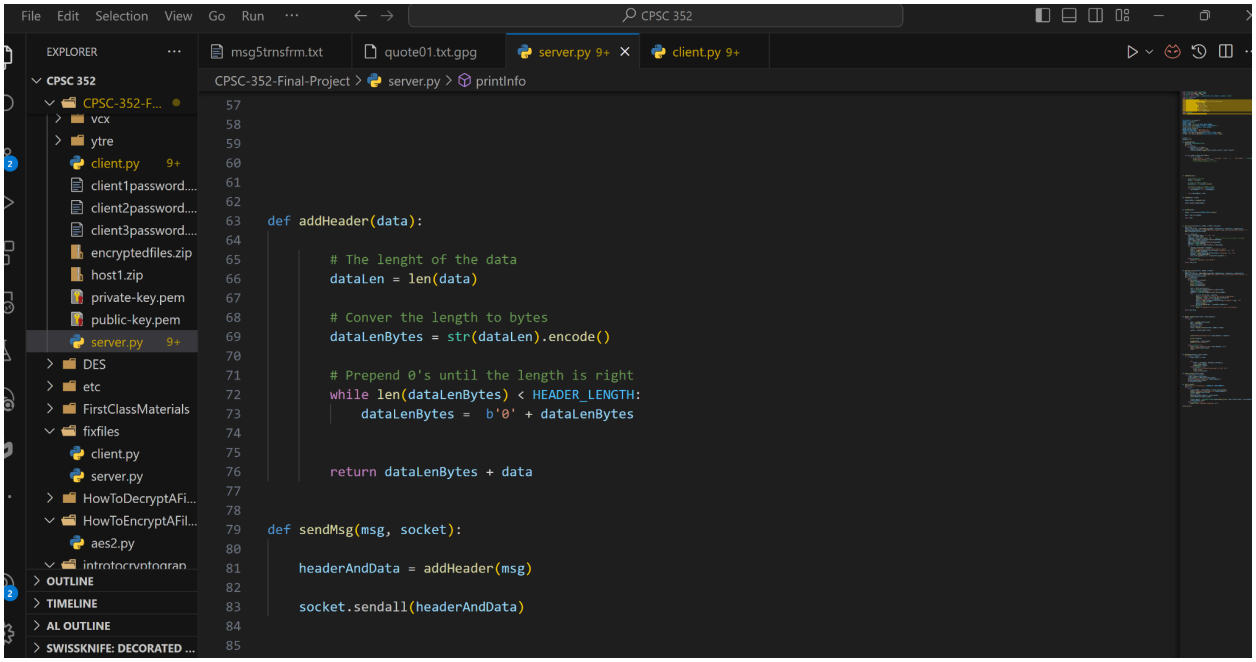

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > server.py > ...
CPSC 352
CPSC-352-F...
vcx
ytre
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
host1.zip
private-key.pem
public-key.pem
server.py 9+
DES
etc
FirstClassMaterials
fixfiles
client.py
server.py
HowToDecryptAFi...
HowToEncryptAFil...
aes2.py
introtocryptograp...
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...
Import "Cryptodome.Signature.pkcs1_15" could not be resolved Pylance(reportMissingImports)
View Problem (Alt+F8) Quick Fix... (Ctrl+.)
1 from collections import defaultdict
2 from os import listdir, mkdir, path
3 from pickle import dumps, loads
4 from
5 from
6 impor
7 from
8 from Cryptodome.Signature.pkcs1_15 import PKCS115_SigScheme
9 from Cryptodome.Hash import SHA256
10 from Cryptodome.Cipher import AES
11 from Crypto.Util.Padding import pad
12 from Crypto.Util.Padding import unpad
13 from Cryptodome.Cipher import AES
14 from Cryptodome.PublicKey import RSA
15 from Cryptodome.Cipher import PKCS1_OAEP
16 import binascii
17 import Cryptodome.Signature.pkcs1_15
18
19
20
21
22 PORT_NUMBER = int(argv[1])
23 SERVER_IP = argv[2]
24 HEADER_LENGTH = 5
25 server_socket = socket(AF_INET, SOCK_STREAM)
26 server_socket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
27 server_socket.bind((SERVER_IP, PORT_NUMBER))
28 server_socket.listen(5)
29 PUBLIC_KEY_FILE_NAME = "public-key.pem"

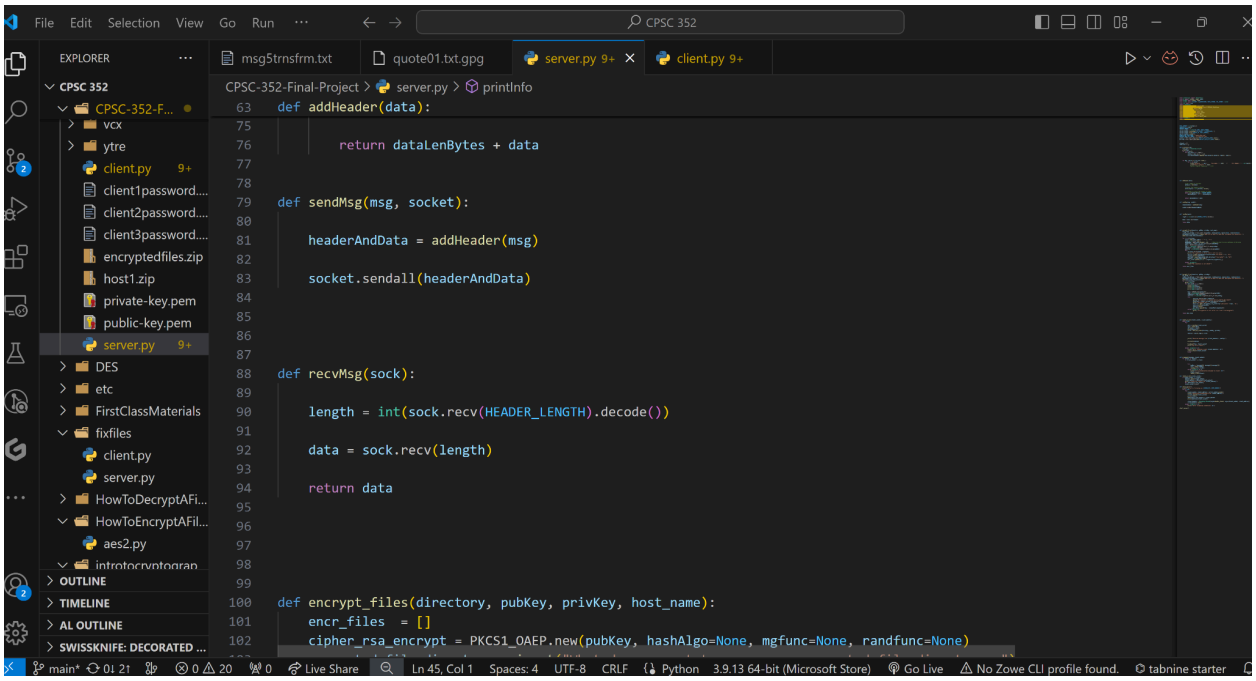


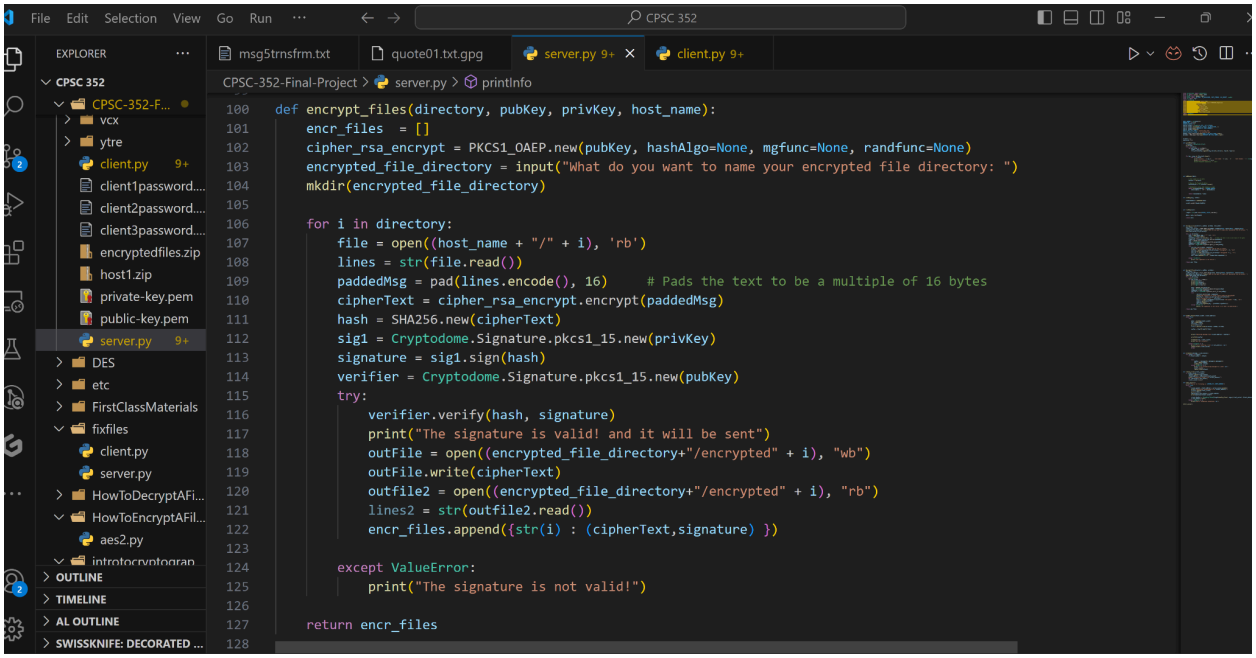
```
pubKey = RSA.import_key(open(PUBLIC_KEY_FILE_NAME).read())
privKey = RSA.import_key(open(PRIVATE_KEY_FILE_NAME).read())


clients = []
addresses = {}

def printInfo(tup):
    file_dict = defaultdict(list)
    print(tup)
    for i in tup[2]:
        for key,value in i.items():
            newKey = key.split('.')[0]
            file_dict[newKey].append((key,value[0],value[1], tup[0], tup[1]))



    for key, value in file_dict.items():
            for i in value:
                print("Keyword: " + key + "    File name: " + i[0]    +"      Port Number: " +  str(tup[0]) + "   Dom
                print("File contents: ", i[1])
                #print("Digital Signature: ", i[2])
```

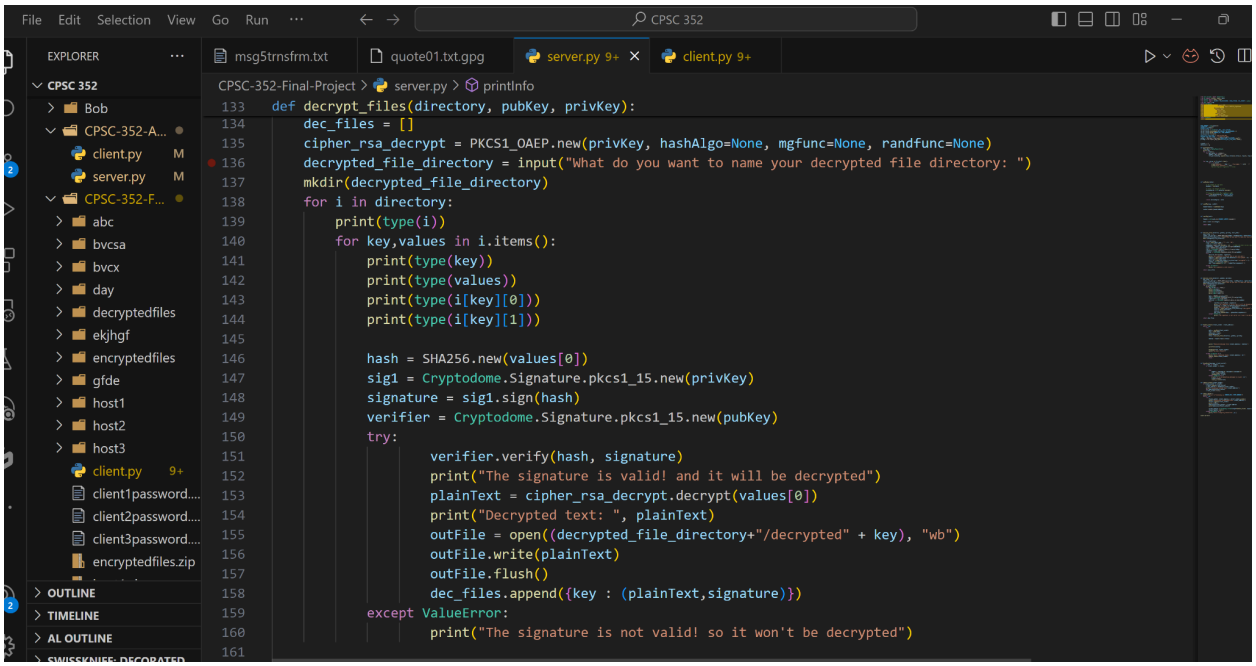


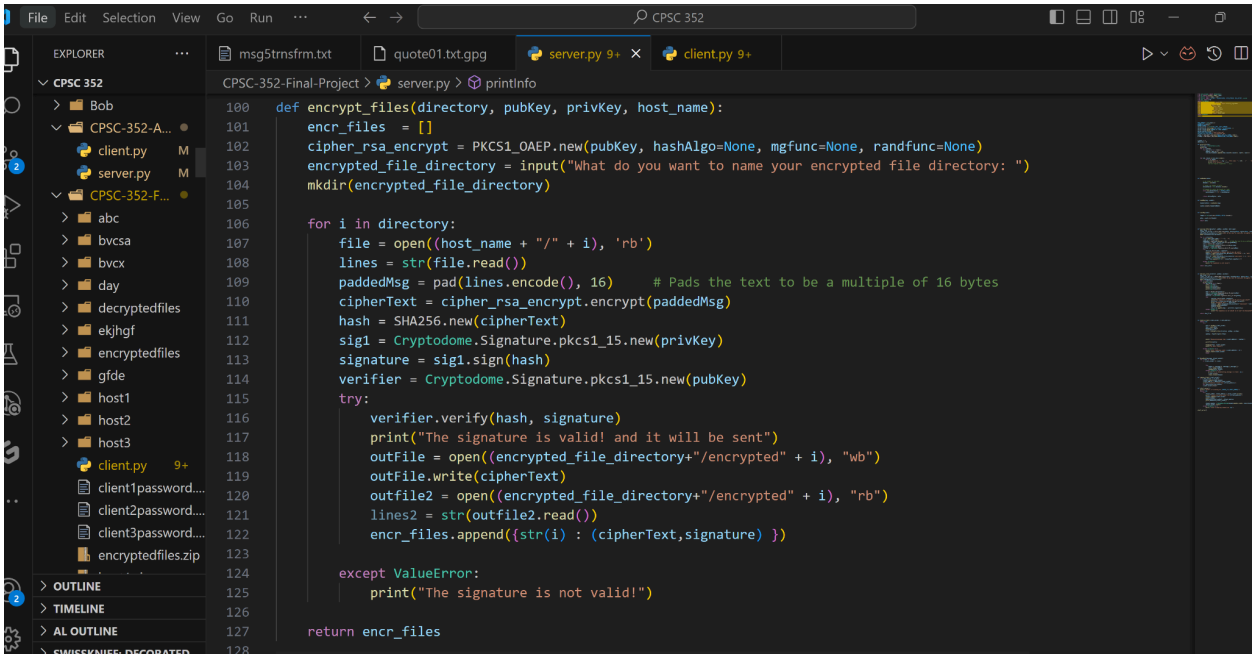
```
pubKey = RSA.import_key(open(PUBLIC_KEY_FILE_NAME).read())
privKey = RSA.import_key(open(PRIVATE_KEY_FILE_NAME).read())


clients = []
addresses = {}

def printInfo(tup):
    file_dict = defaultdict(list)
    print(tup)
    for i in tup[2]:
        for key,value in i.items():
            newKey = key.split('.')[0]
            file_dict[newKey].append((key,value[0],value[1], tup[0], tup[1]))



    for key, value in file_dict.items():
            for i in value:
                print("Keyword: " + key + "    File name: " + i[0]    +"      Port Number: " +  str(tup[0]) + "   Dom
                print("File contents: ", i[1])
                #print("Digital Signature: ", i[2])
```

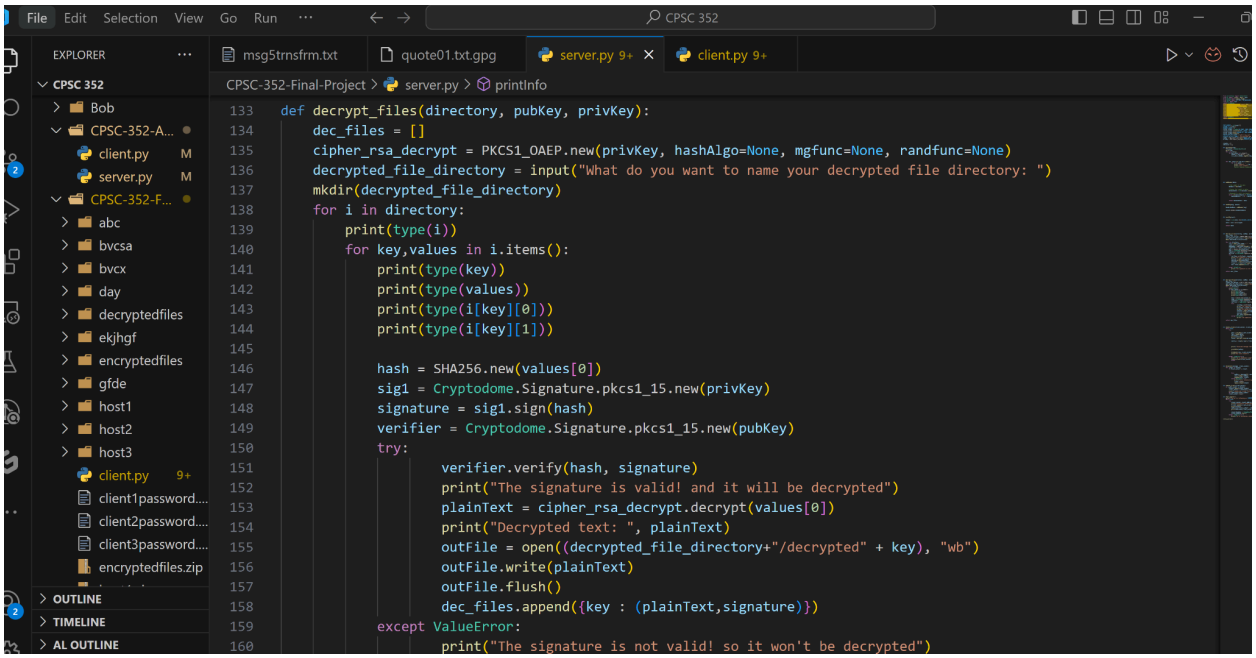

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > server.py > printInfo
CPSC 352
CPSC-352-F...
vcx
ytre
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
host1.zip
private-key.pem
public-key.pem
server.py 9+
DES
etc
FirstClassMaterials
fixfiles
client.py
server.py
HowToDecryptAFi...
HowToEncryptAFil...
aes2.py
introtocryptograp
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...
57
58
59
60
61
62
63 def addHeader(data):
64
65 # The lenght of the data
66 dataLen = len(data)
67
68 # Conver the length to bytes
69 dataLenBytes = str(dataLen).encode()
70
71 # Prepend 0's until the length is right
72 while len(dataLenBytes) < HEADER_LENGTH:
73 dataLenBytes = b'0' + dataLenBytes
74
75
76 return dataLenBytes + data
77
78
79 def sendMsg(msg, socket):
80
81 headerAndData = addHeader(msg)
82
83 socket.sendall(headerAndData)
84
85

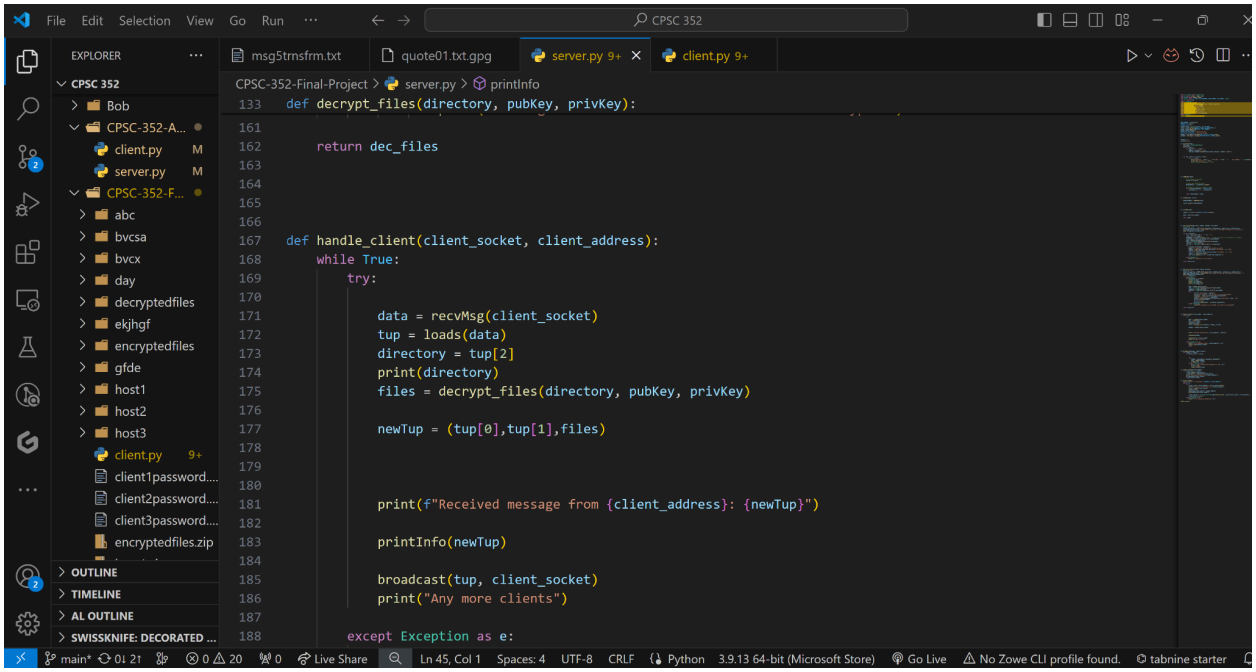

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > server.py > printInfo
CPSC 352
CPSC-352-F...
vcx
ytre
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
host1.zip
private-key.pem
public-key.pem
server.py 9+
DES
etc
FirstClassMaterials
fixfiles
client.py
server.py
HowToDecryptAFi...
HowToEncryptAFil...
aes2.py
introtocryptograp
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...
63 def addHeader(data):
75
76 return dataLenBytes + data
77
78
79 def sendMsg(msg, socket):
80
81 headerAndData = addHeader(msg)
82
83 socket.sendall(headerAndData)
84
85
86
87
88 def recvMsg(sock):
89
90 length = int(sock.recv(HEADER_LENGTH).decode())
91
92 data = sock.recv(length)
93
94 return data
95
96
97
98
99
100 def encrypt_files(directory, pubKey, privKey, host_name):
101 encr_files = []
102 cipher_rsa_encrypt = PKCS1_OAEP.new(pubKey, hashAlgo=None, mgfunc=None, randfunc=None)
main* 0↓ 2↑
0 20
0
Live Share
Ln 45, Col 1
Spaces: 4
UTF-8
CRLF
Python
3.9.13 64-bit (Microsoft Store)
Go Live
No Zowe CLI profile found.
tabnine starter


File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > server.py > printInfo
CPSC 352
CPSC-352-F...
vcx
ytre
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
host1.zip
private-key.pem
public-key.pem
server.py 9+
DES
etc
FirstClassMaterials
fixfiles
client.py
server.py
HowToDecryptAFi...
HowToEncryptAFil...
aes2.py
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...
def encrypt_files(directory, pubKey, privKey, host_name):
encr_files = []
cipher_rsa_encrypt = PKCS1_OAEP.new(pubKey, hashAlgo=None, mgfunc=None, randfunc=None)
encrypted_file_directory = input("What do you want to name your encrypted file directory: ")
mkdir(encrypted_file_directory)
for i in directory:
file = open((host_name + "/" + i), 'rb')
lines = str(file.read())
paddedMsg = pad(lines.encode(), 16) # Pads the text to be a multiple of 16 bytes
cipherText = cipher_rsa_encrypt.encrypt(paddedMsg)
hash = SHA256.new(cipherText)
sig1 = Cryptodome.Signature.pkcs1_15.new(privKey)
signature = sig1.sign(hash)
verifier = Cryptodome.Signature.pkcs1_15.new(pubKey)
try:
verifier.verify(hash, signature)
print("The signature is valid! and it will be sent")
outFile = open((encrypted_file_directory+"/encrypted" + i), "wb")
outFile.write(cipherText)
outfile2 = open((encrypted_file_directory+"/encrypted" + i), "rb")
lines2 = str(outfile2.read())
encr_files.append({str(i) : (cipherText,signature) })
except ValueError:
print("The signature is not valid!")
return encr_files

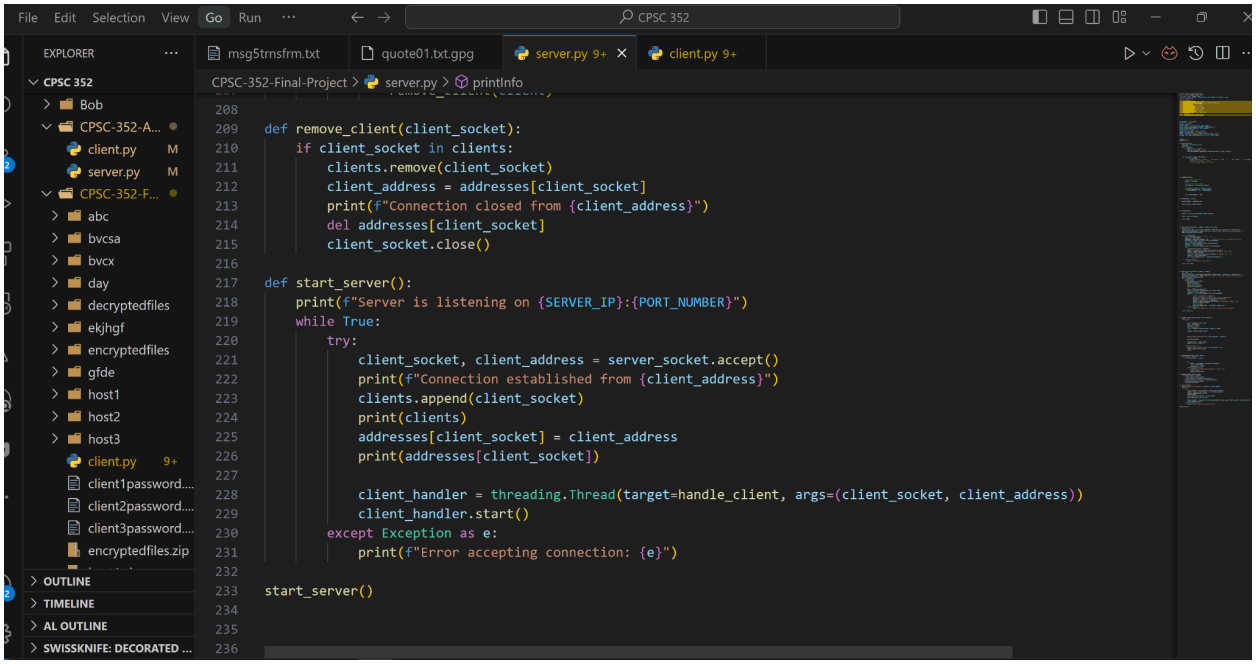

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > server.py > printInfo
CPSC 352
Bob
CPSC-352-A...
client.py M
server.py M
CPSC-352-F...
abc
bvcsa
bvcx
day
decryptedfiles
ekjhgf
encryptedfiles
gfde
host1
host2
host3
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
OUTLINE
TIMELINE
AL OUTLINE
def decrypt_files(directory, pubKey, privKey):
dec_files = []
cipher_rsa_decrypt = PKCS1_OAEP.new(privKey, hashAlgo=None, mgfunc=None, randfunc=None)
decrypted_file_directory = input("What do you want to name your decrypted file directory: ")
mkdir(decrypted_file_directory)
for i in directory:
print(type(i))
for key,values in i.items():
print(type(key))
print(type(values))
print(type(i[key][0]))
print(type(i[key][1]))
hash = SHA256.new(values[0])
sig1 = Cryptodome.Signature.pkcs1_15.new(privKey)
signature = sig1.sign(hash)
verifier = Cryptodome.Signature.pkcs1_15.new(pubKey)
try:
verifier.verify(hash, signature)
print("The signature is valid! and it will be decrypted")
plainText = cipher_rsa_decrypt.decrypt(values[0])
print("Decrypted text: ", plainText)
outFile = open((decrypted_file_directory+"/decrypted" + key), "wb")
outFile.write(plainText)
outFile.flush()
dec_files.append({key : (plainText,signature)})
except ValueError:
print("The signature is not valid! so it won't be decrypted")


File Edit Selection View Go Run ···
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > server.py > printInfo
CPSC 352
Bob
CPSC-352-A...
client.py M
server.py M
CPSC-352-F...
abc
bvcsa
bvcx
day
decryptedfiles
ekjhgf
encryptedfiles
gfde
host1
host2
host3
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
OUTLINE
TIMELINE
AL OUTLINE
100 def encrypt_files(directory, pubKey, privKey, host_name):
101 encr_files = []
102 cipher_rsa_encrypt = PKCS1_OAEP.new(pubKey, hashAlgo=None, mgfunc=None, randfunc=None)
103 encrypted_file_directory = input("What do you want to name your encrypted file directory: ")
104 mkdir(encrypted_file_directory)
105
106 for i in directory:
107 file = open((host_name + "/" + i), 'rb')
108 lines = str(file.read())
109 paddedMsg = pad(lines.encode(), 16) # Pads the text to be a multiple of 16 bytes
110 cipherText = cipher_rsa_encrypt.encrypt(paddedMsg)
111 hash = SHA256.new(cipherText)
112 sig1 = Cryptodome.Signature.pkcs1_15.new(privKey)
113 signature = sig1.sign(hash)
114 verifier = Cryptodome.Signature.pkcs1_15.new(pubKey)
115 try:
116 verifier.verify(hash, signature)
117 print("The signature is valid! and it will be sent")
118 outFile = open((encrypted_file_directory+"/encrypted" + i), "wb")
119 outFile.write(cipherText)
120 outfile2 = open((encrypted_file_directory+"/encrypted" + i), "rb")
121 lines2 = str(outfile2.read())
122 encr_files.append({str(i) : (cipherText,signature) })
123
124 except ValueError:
125 print("The signature is not valid!")
126
127 return encr_files

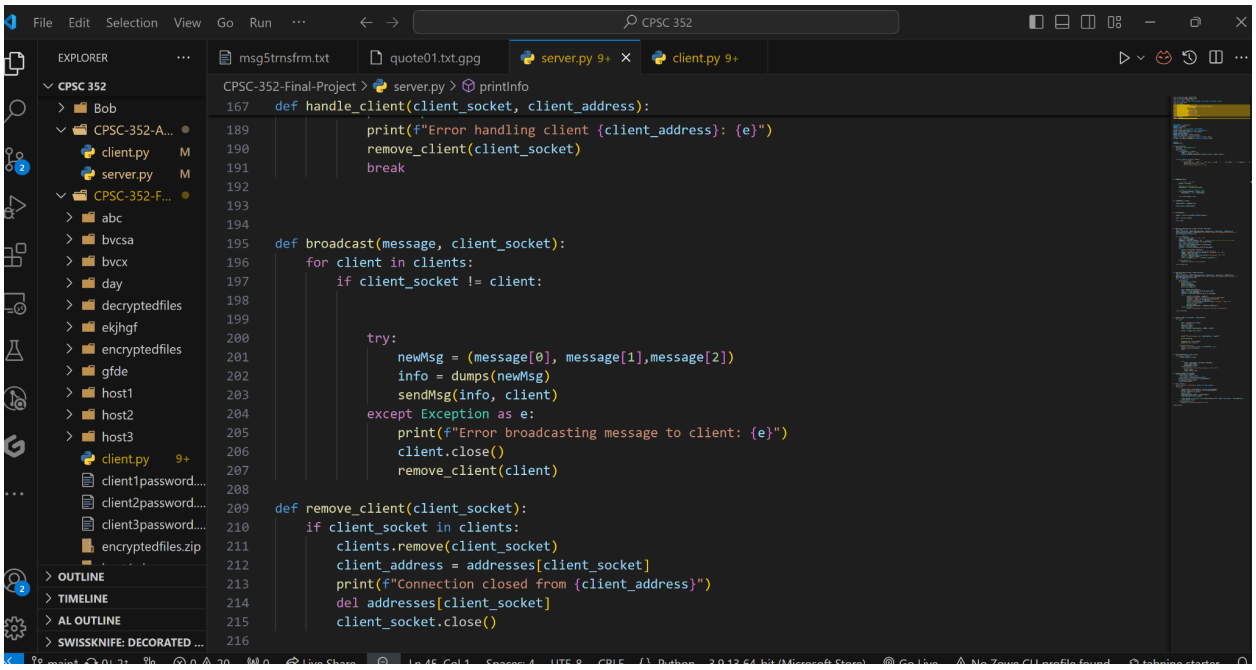

File Edit Selection View Go Run ···
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > server.py > printInfo
CPSC 352
Bob
CPSC-352-A...
client.py M
server.py M
CPSC-352-F...
abc
bvcsa
bvcx
day
decryptedfiles
ekjhgf
encryptedfiles
gfde
host1
host2
host3
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
OUTLINE
TIMELINE
AL OUTLINE
133 def decrypt_files(directory, pubKey, privKey):
134 dec_files = []
135 cipher_rsa_decrypt = PKCS1_OAEP.new(privKey, hashAlgo=None, mgfunc=None, randfunc=None)
136 decrypted_file_directory = input("What do you want to name your decrypted file directory: ")
137 mkdir(decrypted_file_directory)
138 for i in directory:
139 print(type(i))
140 for key,values in i.items():
141 print(type(key))
142 print(type(values))
143 print(type(i[key][0]))
144 print(type(i[key][1]))
145
146 hash = SHA256.new(values[0])
147 sig1 = Cryptodome.Signature.pkcs1_15.new(privKey)
148 signature = sig1.sign(hash)
149 verifier = Cryptodome.Signature.pkcs1_15.new(pubKey)
150 try:
151 verifier.verify(hash, signature)
152 print("The signature is valid! and it will be decrypted")
153 plainText = cipher_rsa_decrypt.decrypt(values[0])
154 print("Decrypted text: ", plainText)
155 outFile = open((decrypted_file_directory+"/decrypted" + key), "wb")
156 outFile.write(plainText)
157 outFile.flush()
158 dec_files.append({key : (plainText,signature)})
159 except ValueError:
160 print("The signature is not valid! so it won't be decrypted")


File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > server.py > printInfo
CPSC 352
Bob
CPSC-352-A...
client.py M
server.py M
CPSC-352-F...
abc
bvcsa
bvcx
day
decryptedfiles
ekjhgf
encryptedfiles
gfde
host1
host2
host3
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...
def decrypt_files(directory, pubKey, privKey):
return dec_files
def handle_client(client_socket, client_address):
while True:
try:
data = recvMsg(client_socket)
tup = loads(data)
directory = tup[2]
print(directory)
files = decrypt_files(directory, pubKey, privKey)
newTup = (tup[0],tup[1],files)
print(f"Received message from {client_address}: {newTup}")
printInfo(newTup)
broadcast(tup, client_socket)
print("Any more clients")
except Exception as e:
main* 01 21
0 20
0
Live Share
Ln 45, Col 1
Spaces: 4
UTF-8
CRLF
Python
3.9.13 64-bit (Microsoft Store)
Go Live
No Zowe CLI profile found.
tabnine starter

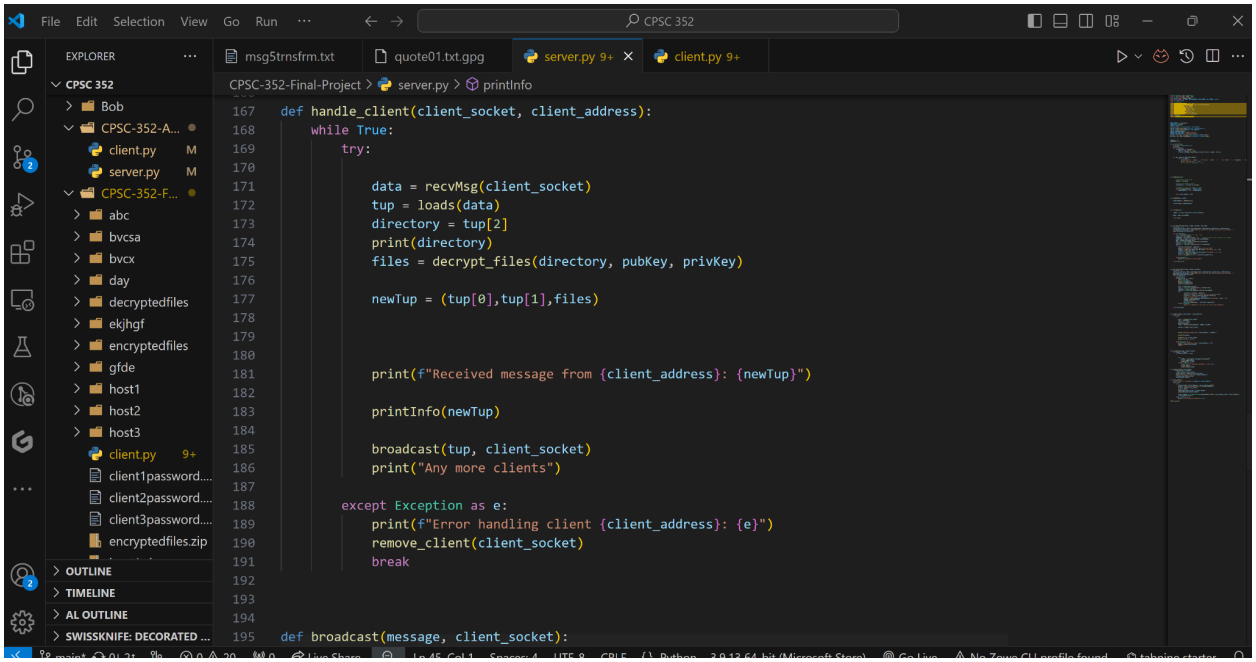

Bob
CPSC-352-A...
client.py M
server.py M
CPSC-352-F...
abc
bvcsa
bvcx
day
decryptedfiles
ekjhgf
encryptedfiles
gfde
host1
host2
host3
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
OUTLINE
TIMELINE
AL OUTLINE
def handle_client(client_socket, client_address):
while True:
try:
data = recvMsg(client_socket)
tup = loads(data)
directory = tup[2]
print(directory)
files = decrypt_files(directory, pubKey, privKey)
newTup = (tup[0],tup[1],files)
print(f"Received message from {client_address}: {newTup}")
printInfo(newTup)
broadcast(tup, client_socket)
print("Any more clients")
except Exception as e:
print(f"Error handling client {client_address}: {e}")
remove_client(client_socket)
break

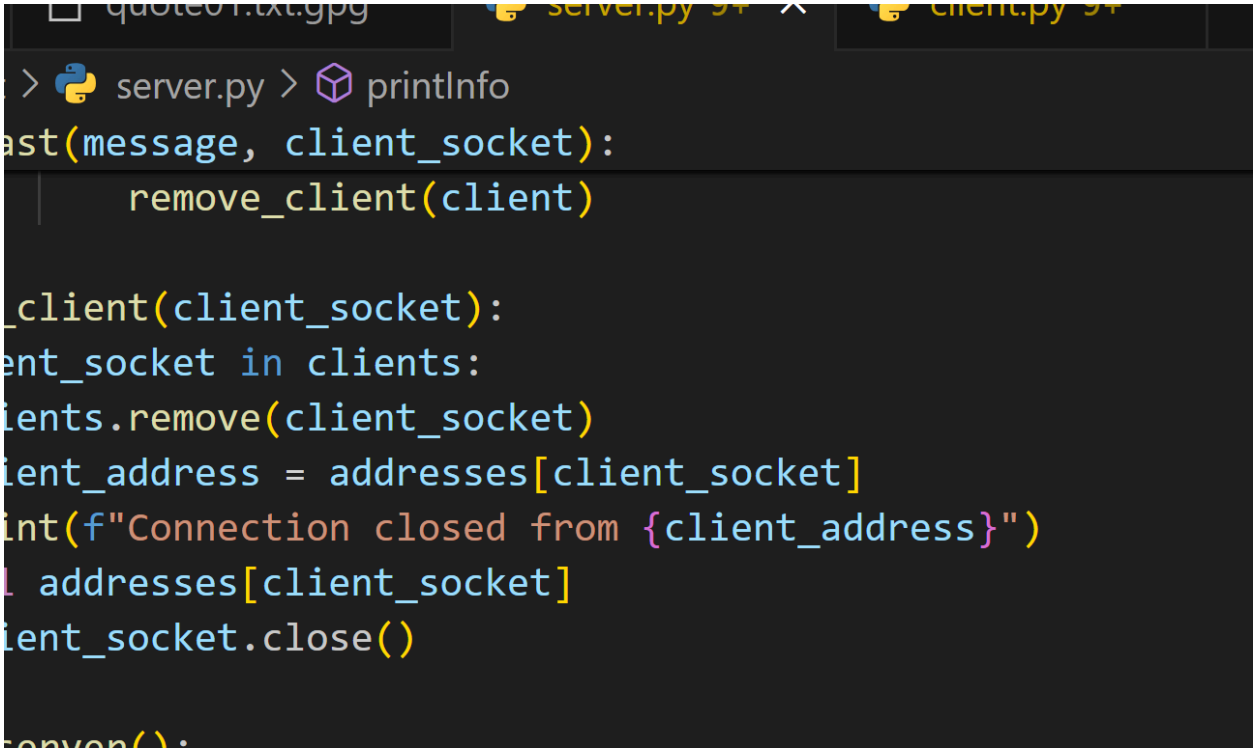


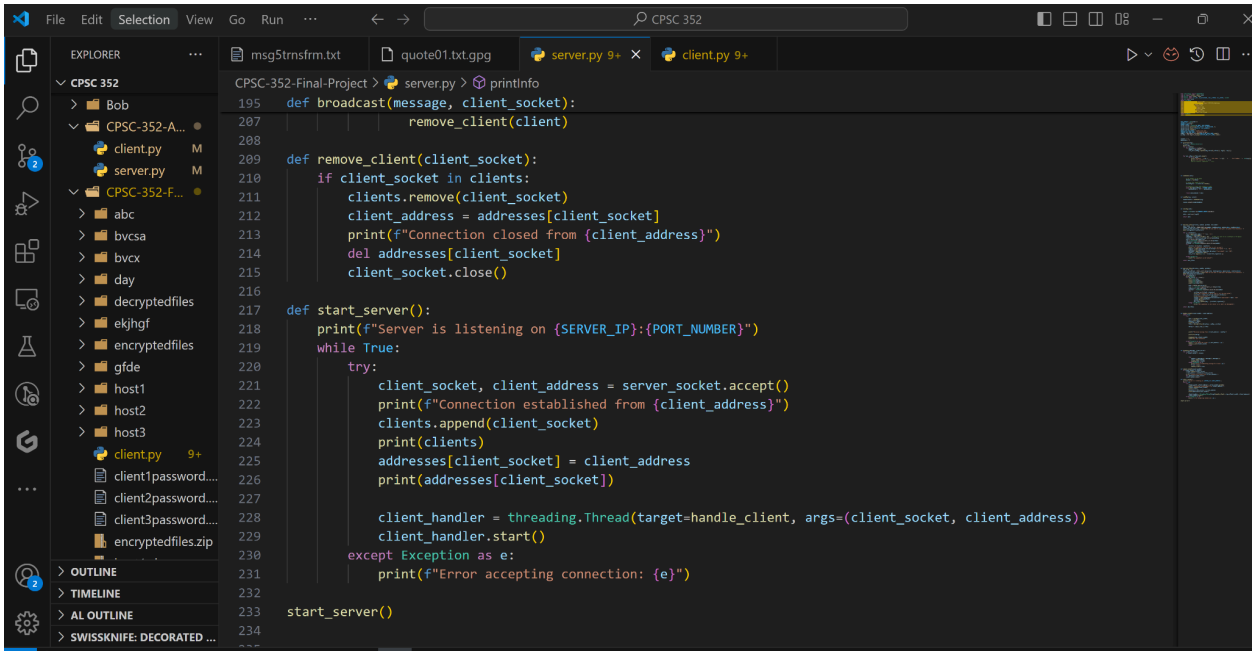
```
def remove_client(client_socket):
    if client_socket in clients:
        clients.remove(client_socket)
        client_address = addresses[client_socket]
        print(f"Connection closed from {client_address}")
        del addresses[client_socket]
        client_socket.close()

def start_server():
    print(f"Server is listening on {SERVER_IP}:{PORT_NUMBER}")
    while True:
        try:
            client_socket, client_address = server_socket.accept()
            print(f"Connection established from {client_address}")
            clients.append(client_socket)
            print(clients)
            addresses[client_socket] = client_address
            print(addresses[client_socket])

            client_handler = threading.Thread(target=handle_client, args=(client_socket, client_address))
            client_handler.start()
        except Exception as e:
            print(f"Error accepting connection: {e}")

start_server()
```

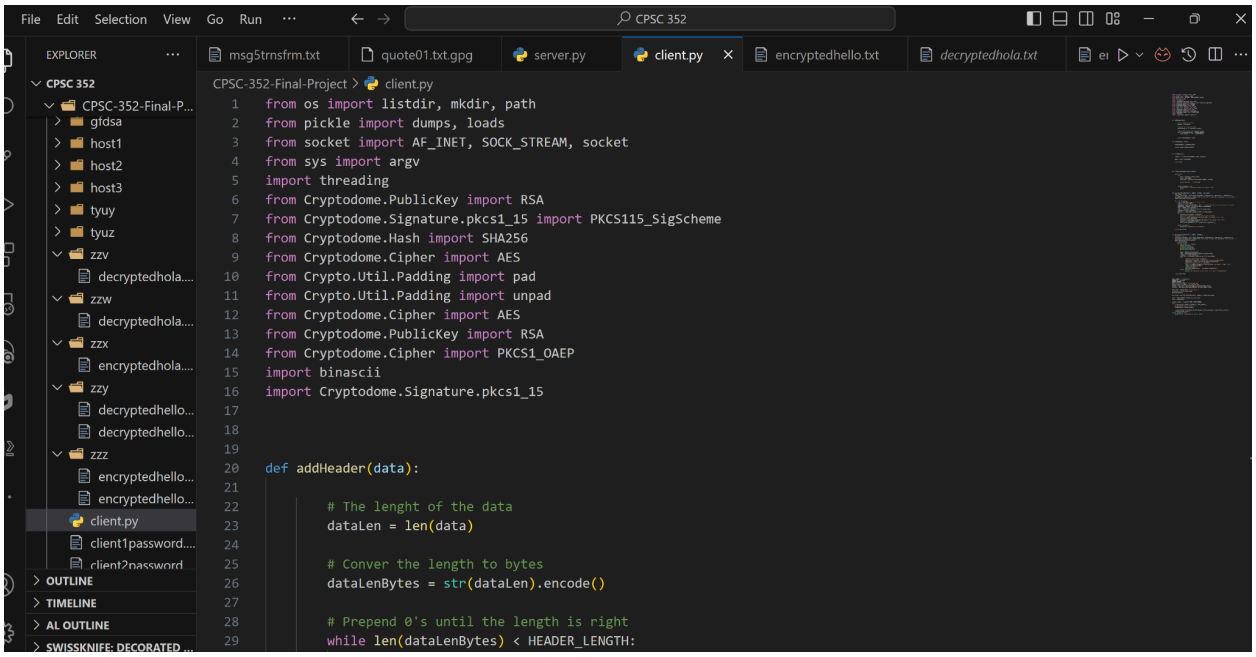


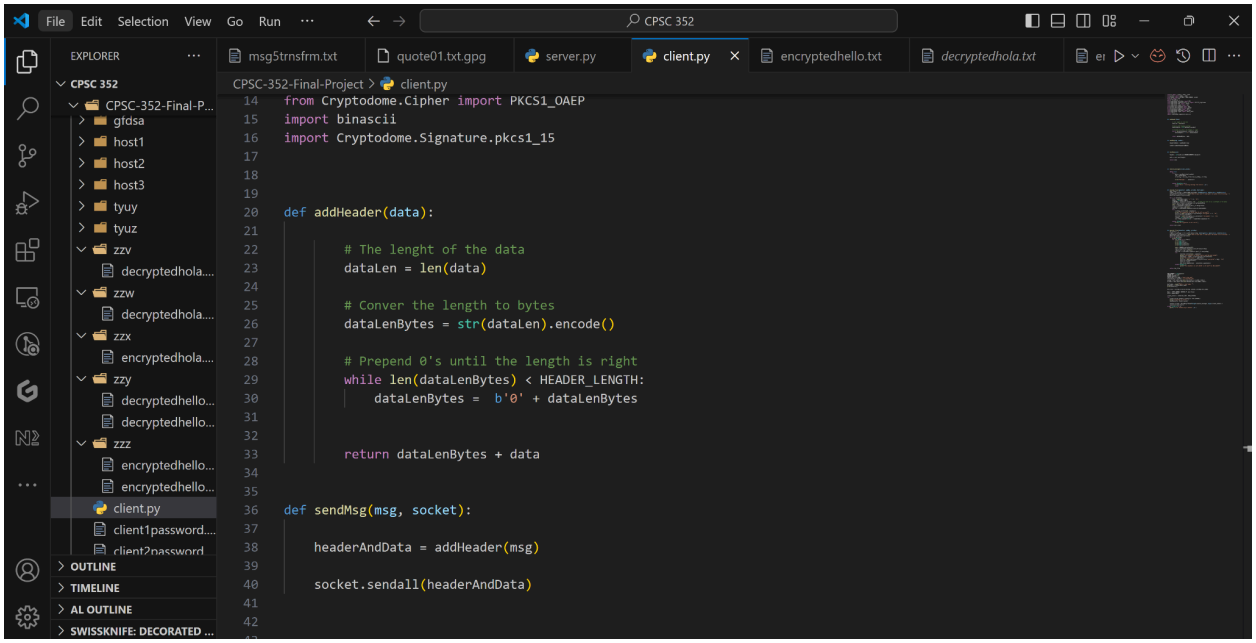
```
def handle_client(client_socket, client_address):
                print(f"Error handling client {client_address}: {e}")
                remove_client(client_socket)
                break


def broadcast(message, client_socket):
    for client in clients:
        if client_socket != client:


            try:
                newMsg = (message[0], message[1],message[2])
                info = dumps(newMsg)
                sendMsg(info, client)
            except Exception as e:
                print(f"Error broadcasting message to client: {e}")
                client.close()
                remove_client(client)

def remove_client(client_socket):
    if client_socket in clients:
        clients.remove(client_socket)
        client_address = addresses[client_socket]
        print(f"Connection closed from {client_address}")
        del addresses[client_socket]
        client_socket.close()
```

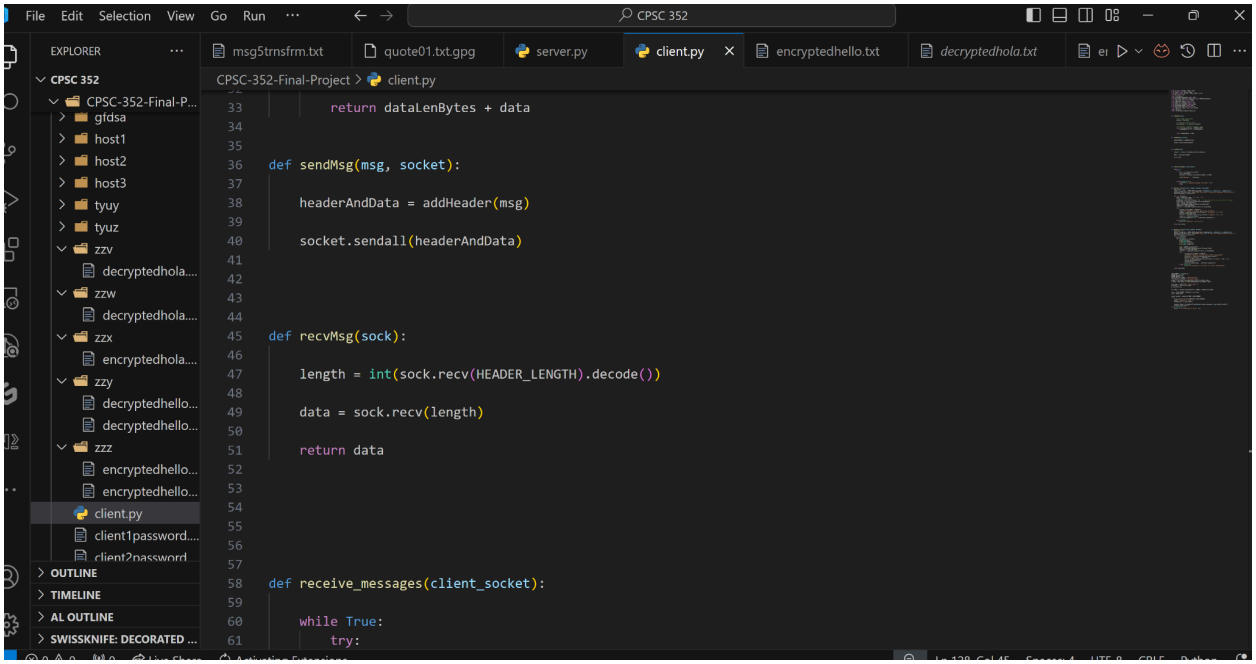

File Edit Selection View Go Run
CPSC 352
EXPLORER
CPSC 352
Bob
CPSC-352-A...
client.py M
server.py M
CPSC-352-F...
abc
bvcsa
bvcx
day
decryptedfiles
ekjhgf
encryptedfiles
gfde
host1
host2
host3
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > server.py > printInfo
def handle_client(client_socket, client_address):
while True:
try:
data = recvMsg(client_socket)
tup = loads(data)
directory = tup[2]
print(directory)
files = decrypt_files(directory, pubKey, privKey)
newTup = (tup[0],tup[1],files)
print(f"Received message from {client_address}: {newTup}")
printInfo(newTup)
broadcast(tup, client_socket)
print("Any more clients")
except Exception as e:
print(f"Error handling client {client_address}: {e}")
remove_client(client_socket)
break
def broadcast(message, client_socket):


server.py > printInfo
ast(message, client_socket):
remove_client(client)
_client(client_socket):
ent_socket in clients:
ients.remove(client_socket)
ient_address = addresses[client_socket]
int(f"Connection closed from {client_address}")
addresses[client_socket]
ient_socket.close()

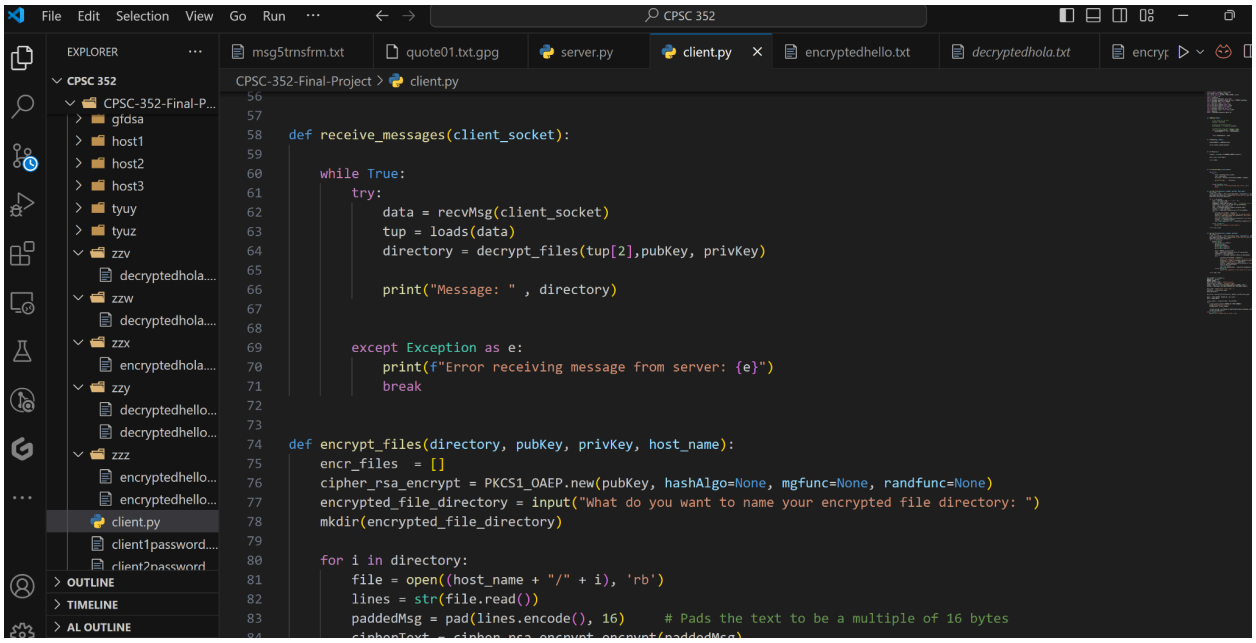

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py 9+
client.py 9+
CPSC-352-Final-Project > server.py > printInfo
CPSC 352
Bob
CPSC-352-A...
client.py M
server.py M
CPSC-352-F...
abc
bvcsa
bvcx
day
decryptedfiles
ekjhgf
encryptedfiles
gfde
host1
host2
host3
client.py 9+
client1password....
client2password....
client3password....
encryptedfiles.zip
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...
195 def broadcast(message, client_socket):
207 remove_client(client)
208
209 def remove_client(client_socket):
210 if client_socket in clients:
211 clients.remove(client_socket)
212 client_address = addresses[client_socket]
213 print(f"Connection closed from {client_address}")
214 del addresses[client_socket]
215 client_socket.close()
216
217 def start_server():
218 print(f"Server is listening on {SERVER_IP}:{PORT_NUMBER}")
219 while True:
220 try:
221 client_socket, client_address = server_socket.accept()
222 print(f"Connection established from {client_address}")
223 clients.append(client_socket)
224 print(clients)
225 addresses[client_socket] = client_address
226 print(addresses[client_socket])
227
228 client_handler = threading.Thread(target=handle_client, args=(client_socket, client_address))
229 client_handler.start()
230 except Exception as e:
231 print(f"Error accepting connection: {e}")
232
233 start_server()
234


File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py
client.py
encryptedhello.txt
decryptedhola.txt
CPSC-352-Final-Project > client.py
CPSC 352
CPSC-352-Final-P...
gfdsa
host1
host2
host3
tyuy
tyuz
zzv
decryptedhola....
zzw
decryptedhola....
zzx
encryptedhola....
zzy
decryptedhello...
decryptedhello...
zzz
encryptedhello...
encryptedhello...
client.py
client1password....
client2password
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...
1 from os import listdir, mkdir, path
2 from pickle import dumps, loads
3 from socket import AF_INET, SOCK_STREAM, socket
4 from sys import argv
5 import threading
6 from Cryptodome.PublicKey import RSA
7 from Cryptodome.Signature.pkcs1_15 import PKCS115_SigScheme
8 from Cryptodome.Hash import SHA256
9 from Cryptodome.Cipher import AES
10 from Crypto.Util.Padding import pad
11 from Crypto.Util.Padding import unpad
12 from Cryptodome.Cipher import AES
13 from Cryptodome.PublicKey import RSA
14 from Cryptodome.Cipher import PKCS1_OAEP
15 import binascii
16 import Cryptodome.Signature.pkcs1_15
17
18
19
20 def addHeader(data):
21
22 # The lenght of the data
23 dataLen = len(data)
24
25 # Conver the length to bytes
26 dataLenBytes = str(dataLen).encode()
27
28 # Prepend 0's until the length is right
29 while len(dataLenBytes) < HEADER_LENGTH:

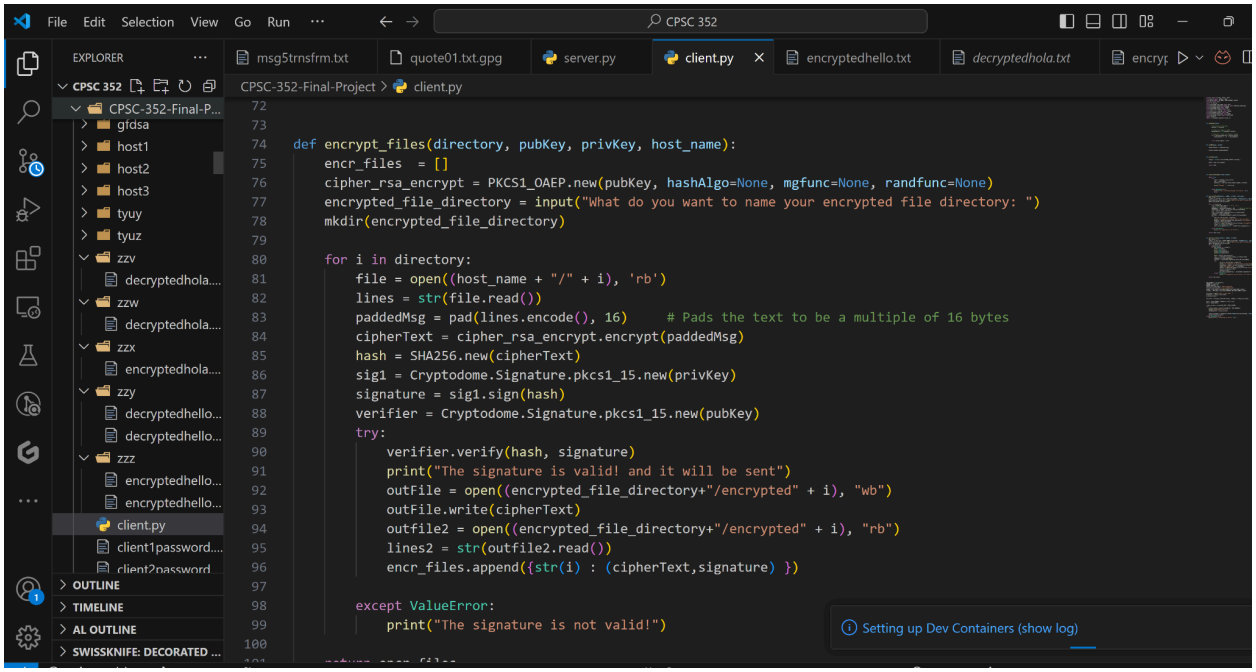

File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py
client.py
encryptedhello.txt
decryptedhola.txt
CPSC-352-Final-Project > client.py
CPSC 352
CPSC-352-Final-P...
gfdsa
host1
host2
host3
tyuy
tyuz
zzv
decryptedhola....
zzw
decryptedhola....
zzx
encryptedhola....
zzy
decryptedhello...
decryptedhello...
zzz
encryptedhello...
encryptedhello...
client.py
client1password....
client2password
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...
14 from Cryptodome.Cipher import PKCS1_OAEP
15 import binascii
16 import Cryptodome.Signature.pkcs1_15
17
18
19
20 def addHeader(data):
21
22 # The lenght of the data
23 dataLen = len(data)
24
25 # Conver the length to bytes
26 dataLenBytes = str(dataLen).encode()
27
28 # Prepend 0's until the length is right
29 while len(dataLenBytes) < HEADER_LENGTH:
30 dataLenBytes = b'0' + dataLenBytes
31
32
33 return dataLenBytes + data
34
35
36 def sendMsg(msg, socket):
37
38 headerAndData = addHeader(msg)
39
40 socket.sendall(headerAndData)
41
42



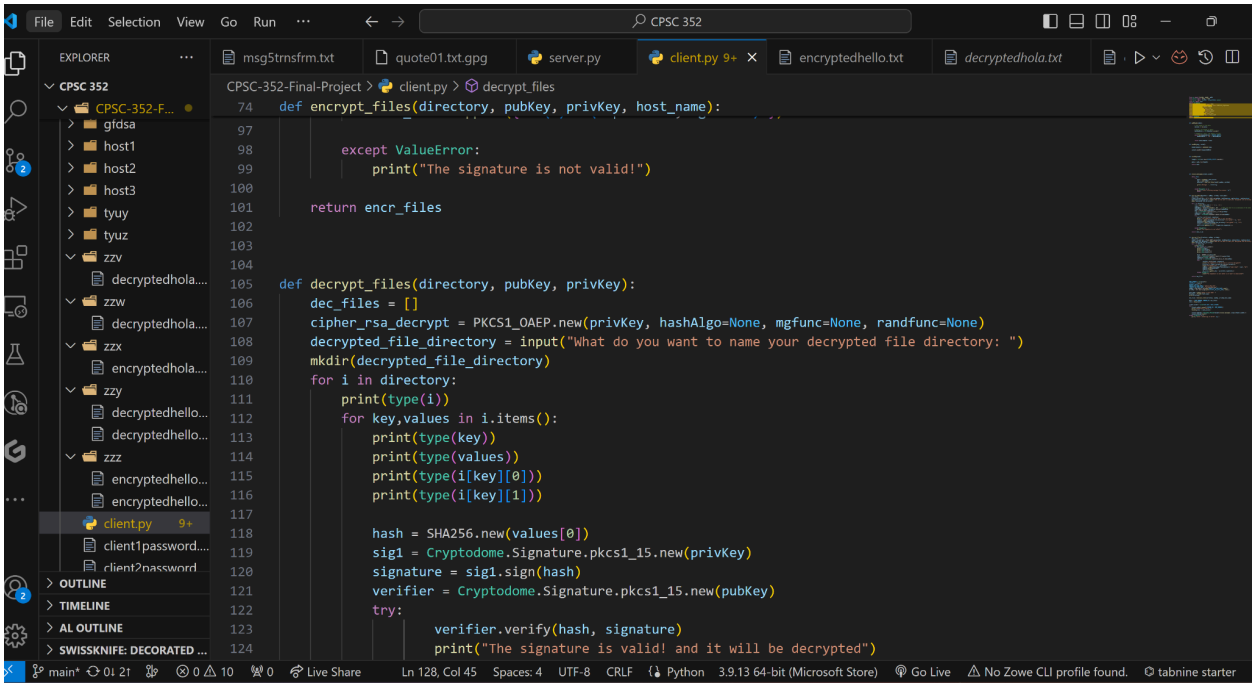
```
        return dataLenBytes + data


def sendMsg(msg, socket):

    headerAndData = addHeader(msg)

    socket.sendall(headerAndData)




def recvMsg(sock):

    length = int(sock.recv(HEADER_LENGTH).decode())

    data = sock.recv(length)

    return data




def receive_messages(client_socket):

    while True:
        try:
```



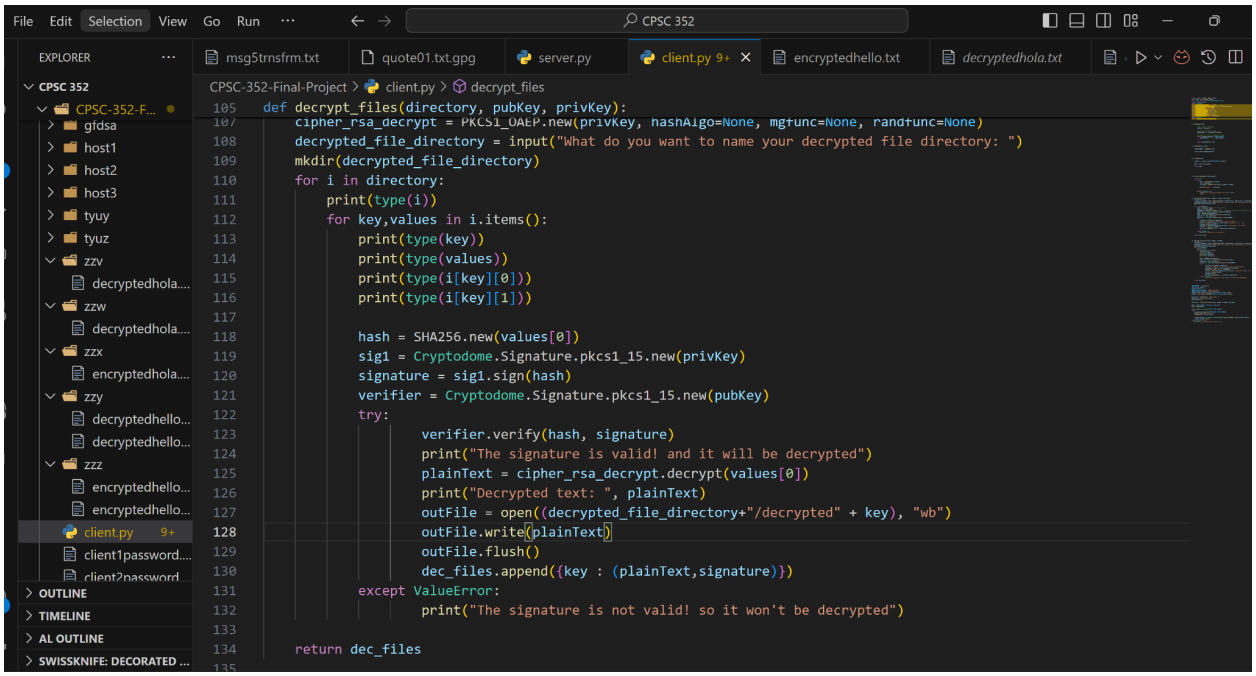
```
def receive_messages(client_socket):

    while True:
        try:
            data = recvMsg(client_socket)
            tup = loads(data)
            directory = decrypt_files(tup[2],pubKey, privKey)

            print("Message: " , directory)


        except Exception as e:
            print(f"Error receiving message from server: {e}")
            break


def encrypt_files(directory, pubKey, privKey, host_name):
    encr_files  = []
    cipher_rsa_encrypt = PKCS1_OAEP.new(pubKey, hashAlgo=None, mgfunc=None, randfunc=None)
    encrypted_file_directory = input("What do you want to name your encrypted file directory: ")
    mkdir(encrypted_file_directory)

    for i in directory:
        file = open((host_name + "/" + i), 'rb')
        lines = str(file.read())
        paddedMsg = pad(lines.encode(), 16)     # Pads the text to be a multiple of 16 bytes
```

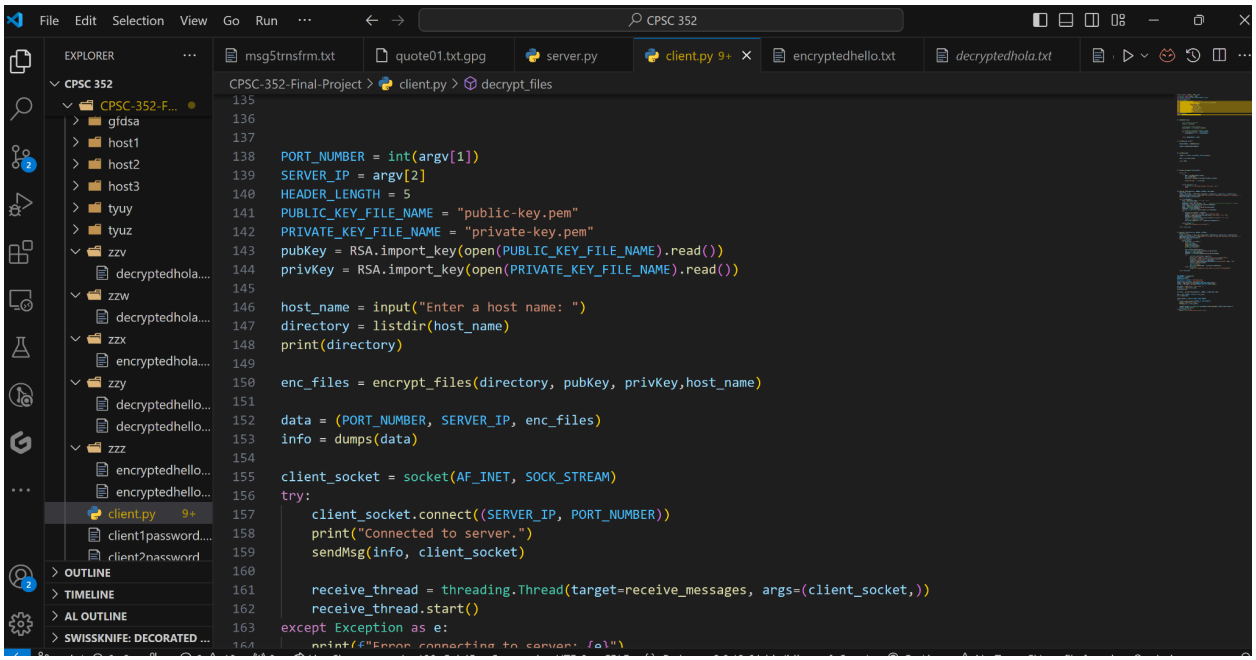


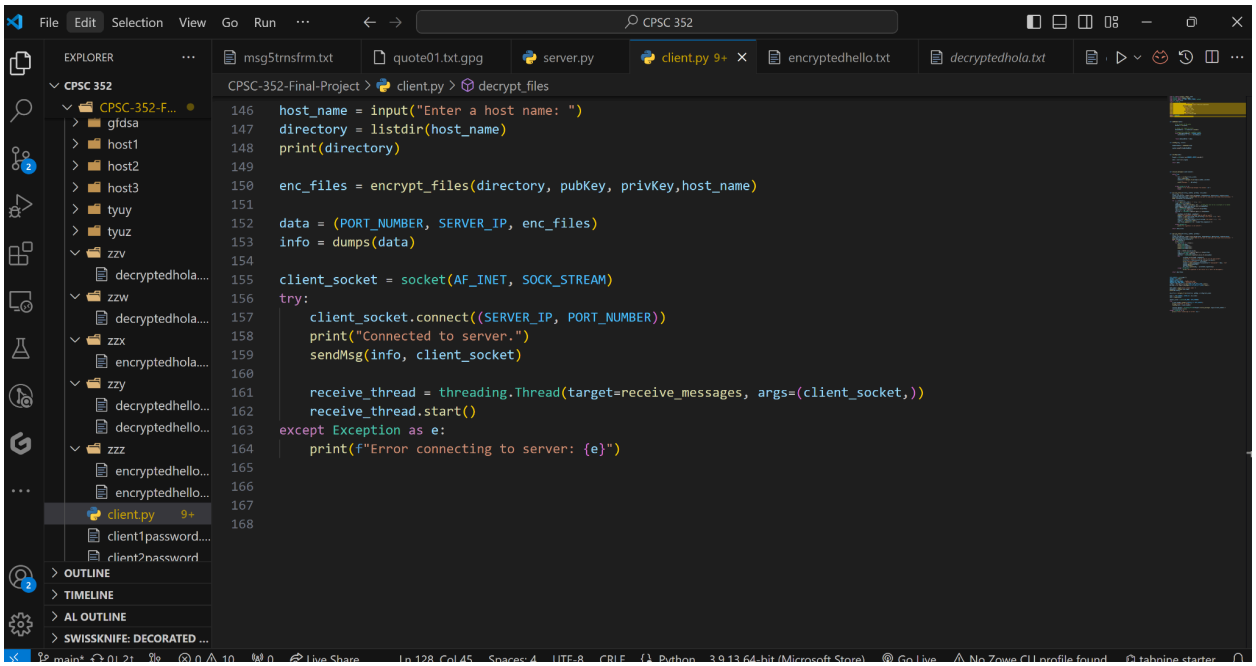
```
def encrypt_files(directory, pubKey, privKey, host_name):
    encr_files  = []
    cipher_rsa_encrypt = PKCS1_OAEP.new(pubKey, hashAlgo=None, mgfunc=None, randfunc=None)
    encrypted_file_directory = input("What do you want to name your encrypted file directory: ")
    mkdir(encrypted_file_directory)

    for i in directory:
        file = open((host_name + "/" + i), 'rb')
        lines = str(file.read())
        paddedMsg = pad(lines.encode(), 16)     # Pads the text to be a multiple of 16 bytes
        cipherText = cipher_rsa_encrypt.encrypt(paddedMsg)
        hash = SHA256.new(cipherText)
        sig1 = Cryptodome.Signature.pkcs1_15.new(privKey)
        signature = sig1.sign(hash)
        verifier = Cryptodome.Signature.pkcs1_15.new(pubKey)
        try:
            verifier.verify(hash, signature)
            print("The signature is valid! and it will be sent")
            outFile = open((encrypted_file_directory+"/encrypted" + i), "wb")
            outFile.write(cipherText)
            outfile2 = open((encrypted_file_directory+"/encrypted" + i), "rb")
            lines2 = str(outfile2.read())
            encr_files.append({str(i) : (cipherText,signature) })

        except ValueError:
            print("The signature is not valid!")
```



```
        except ValueError:
            print("The signature is not valid!")

    return encr_files


def decrypt_files(directory, pubKey, privKey):
    dec_files = []
    cipher_rsa_decrypt = PKCS1_OAEP.new(privKey, hashAlgo=None, mgfunc=None, randfunc=None)
    decrypted_file_directory = input("What do you want to name your decrypted file directory: ")
    mkdir(decrypted_file_directory)
    for i in directory:
        print(type(i))
        for key,values in i.items():
            print(type(key))
            print(type(values))
            print(type(i[key][0]))
            print(type(i[key][1]))

            hash = SHA256.new(values[0])
            sig1 = Cryptodome.Signature.pkcs1_15.new(privKey)
            signature = sig1.sign(hash)
            verifier = Cryptodome.Signature.pkcs1_15.new(pubKey)
            try:
                verifier.verify(hash, signature)
                print("The signature is valid! and it will be decrypted")
```


File Edit Selection View Go Run ···
CPSC 352
EXPLORER
CPSC 352
CPSC-352-F...
gfdsa
host1
host2
host3
tyuy
tyuz
zzv
decryptedhola....
zzw
decryptedhola....
zzx
encryptedhola....
zzy
decryptedhello...
decryptedhello...
zzz
encryptedhello...
encryptedhello...
client.py 9+
client1password....
client2password
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...
msg5trnsfrm.txt
quote01.txt.gpg
server.py
client.py 9+
encryptedhello.txt
decryptedhola.txt
CPSC-352-Final-Project > client.py > decrypt_files
105 def decrypt_files(directory, pubKey, privKey):
107 cipher_rsa_decrypt = PKCS1_OAEP.new(privKey, hashAlgo=None, mgfunc=None, randfunc=None)
108 decrypted_file_directory = input("What do you want to name your decrypted file directory: ")
109 mkdir(decrypted_file_directory)
110 for i in directory:
111 print(type(i))
112 for key,values in i.items():
113 print(type(key))
114 print(type(values))
115 print(type(i[key][0]))
116 print(type(i[key][1]))
117
118 hash = SHA256.new(values[0])
119 sig1 = Cryptodome.Signature.pkcs1_15.new(privKey)
120 signature = sig1.sign(hash)
121 verifier = Cryptodome.Signature.pkcs1_15.new(pubKey)
122 try:
123 verifier.verify(hash, signature)
124 print("The signature is valid! and it will be decrypted")
125 plainText = cipher_rsa_decrypt.decrypt(values[0])
126 print("Decrypted text: ", plainText)
127 outFile = open((decrypted_file_directory+"/decrypted" + key), "wb")
128 outFile.write(plainText)
129 outFile.flush()
130 dec_files.append({key : (plainText,signature)})
131 except ValueError:
132 print("The signature is not valid! so it won't be decrypted")
133
134 return dec_files
135


File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py
client.py 9+
encryptedhello.txt
decryptedhola.txt
CPSC-352-Final-Project > client.py > decrypt_files
PORT_NUMBER = int(argv[1])
SERVER_IP = argv[2]
HEADER_LENGTH = 5
PUBLIC_KEY_FILE_NAME = "public-key.pem"
PRIVATE_KEY_FILE_NAME = "private-key.pem"
pubKey = RSA.import_key(open(PUBLIC_KEY_FILE_NAME).read())
privKey = RSA.import_key(open(PRIVATE_KEY_FILE_NAME).read())
host_name = input("Enter a host name: ")
directory = listdir(host_name)
print(directory)
enc_files = encrypt_files(directory, pubKey, privKey,host_name)
data = (PORT_NUMBER, SERVER_IP, enc_files)
info = dumps(data)
client_socket = socket(AF_INET, SOCK_STREAM)
try:
client_socket.connect((SERVER_IP, PORT_NUMBER))
print("Connected to server.")
sendMsg(info, client_socket)
receive_thread = threading.Thread(target=receive_messages, args=(client_socket,))
receive_thread.start()
except Exception as e:
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...


File Edit Selection View Go Run
CPSC 352
EXPLORER
msg5trnsfrm.txt
quote01.txt.gpg
server.py
client.py 9+
encryptedhello.txt
decryptedhola.txt
CPSC-352-Final-Project > client.py > decrypt_files
host_name = input("Enter a host name: ")
directory = listdir(host_name)
print(directory)
enc_files = encrypt_files(directory, pubKey, privKey,host_name)
data = (PORT_NUMBER, SERVER_IP, enc_files)
info = dumps(data)
client_socket = socket(AF_INET, SOCK_STREAM)
try:
client_socket.connect((SERVER_IP, PORT_NUMBER))
print("Connected to server.")
sendMsg(info, client_socket)
receive_thread = threading.Thread(target=receive_messages, args=(client_socket,))
receive_thread.start()
except Exception as e:
print(f"Error connecting to server: {e}")
OUTLINE
TIMELINE
AL OUTLINE
SWISSKNIFE: DECORATED ...