Week 15 (November 30-December 4, 2020).

Assigned Reading

- Section 12.1: Weierstrass Equations and Elliptic Curves (pp. 213-222)
- Section 12.2: Elliptic Curve Diffie-Hellman (pp. 222-223)
- Section 12.3: Efficiency and Security of Elliptic Curve Cryptography (pp. 223-224)
- Section 12.4: Elliptic Curve Factoring Method (pp. 224-226)

Project 7

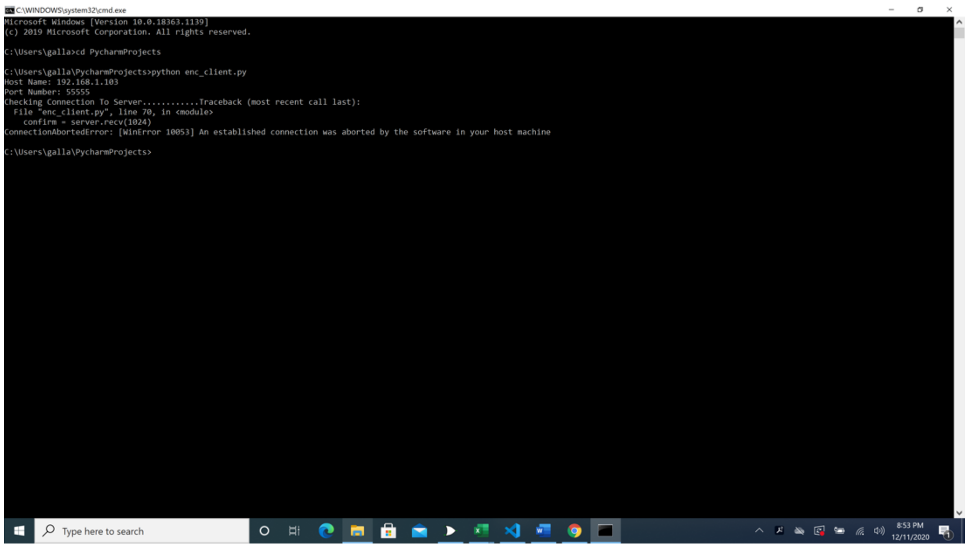
The GitHub site, https://github.com/TomPrograms/Python-Internet-Chat-Room has a fully functional Internet Chat Room written in Python released under the MIT license, which allows use, copying, and modification.

1. Download a copy of this program and run it. What encryption does it use, if any? Is an eavesdropper with access to the network traffic able to read the messages sent to/from the users? Demonstrate.

- **No, the original program did not have any encryption, therefore the eavesdropper is able to read the messages that are sent to and/or from the user if they have access to the network traffic.**

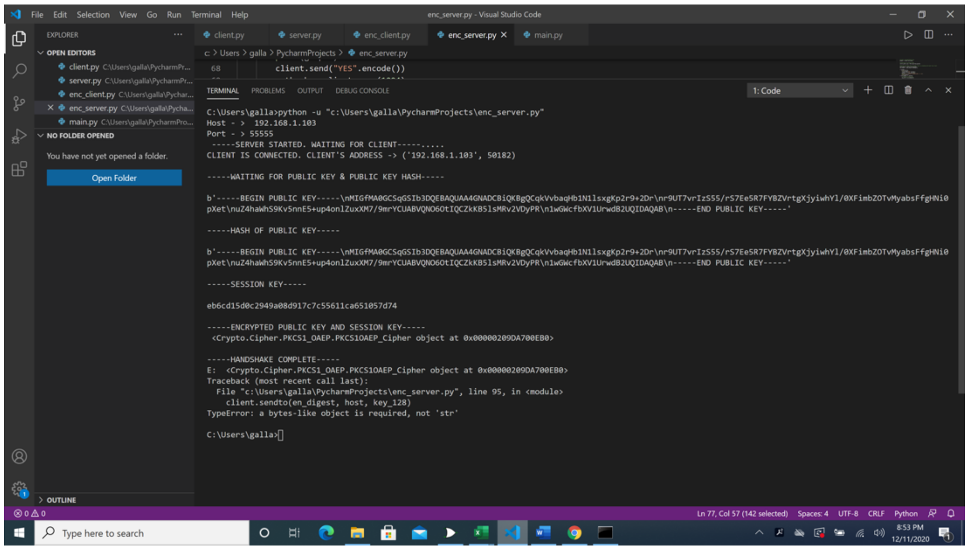
2. Modify the program so that the traffic between server and client is properly encrypted. The program should:
   a. Include a mechanism so that the client and the server can verify the identity of each other.

- **Below we have a screenshot of the client verifying the identity of the server:**



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.1139]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\galla>cd PycharmProjects

C:\Users\galla\PycharmProjects>python enc_client.py
Host Name: 192.168.1.103
Port Number: 55555
Checking Connection To Server............Traceback (most recent call last):
  File "enc_client.py", line 70, in <module>
    confirm = server.recv(1024)
ConnectionAbortedError: [WinError 10053] An established connection was aborted by the software in your host machine

C:\Users\galla\PycharmProjects>
```

- **Below we have a screenshot of the server verifying the identity of the client:**



```
client.send("YES".encode())
```

```
C:\Users\galla>python -u "c:\Users\galla\PycharmProjects\enc_server.py"
Host - >  192.168.1.103
Port - > 55555
 -----SERVER STARTED. WAITING FOR CLIENT-----.....
CLIENT IS CONNECTED. CLIENT'S ADDRESS -> ('192.168.1.103', 50182)

-----WAITING FOR PUBLIC KEY & PUBLIC KEY HASH-----

b'-----BEGIN PUBLIC KEY-----\nMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqkVvbaqHb1N1lsxgKp2r9+2Dr\nr9UT7vrIzS55/rS7Ee5R7FYBZVrtgXjyiwhYl/0XFimbZOTvMyabsFfgHNi0
pXet\nuZ4haWhS9Kv5nnE5+up4onlZuxXM7/9mrYCUABVQNO6OtIQCZkKB5lsMRv2VDyPR\n1wGWcfbXV1UrwdB2UQIDAQAB\n-----END PUBLIC KEY-----'

-----HASH OF PUBLIC KEY-----

b'-----BEGIN PUBLIC KEY-----\nMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqkVvbaqHb1N1lsxgKp2r9+2Dr\nr9UT7vrIzS55/rS7Ee5R7FYBZVrtgXjyiwhYl/0XFimbZOTvMyabsFfgHNi0
pXet\nuZ4haWhS9Kv5nnE5+up4onlZuxXM7/9mrYCUABVQNO6OtIQCZkKB5lsMRv2VDyPR\n1wGWcfbXV1UrwdB2UQIDAQAB\n-----END PUBLIC KEY-----'

-----SESSION KEY-----

eb6cd15d0c2949a08d917c7c55611ca651057d74

-----ENCRYPTED PUBLIC KEY AND SESSION KEY-----
 <Crypto.Cipher.PKCS1_OAEP.PKCS1OAEP_Cipher object at 0x00000209DA700EB0>

-----HANDSHAKE COMPLETE-----
E:  <Crypto.Cipher.PKCS1_OAEP.PKCS1OAEP_Cipher object at 0x00000209DA700EB0>
Traceback (most recent call last):
  File "c:\Users\galla\PycharmProjects\enc_server.py", line 95, in <module>
    client.sendto(en_digest, host, key_128)
TypeError: a bytes-like object is required, not 'str'

C:\Users\galla>
```

File Edit Selection View Go Run Terminal Help
enc_server.py - Visual Studio Code
client.py server.py enc_client.py enc_server.py main.py
c: > Users > galla > PycharmProjects > enc_server.py
EXPLORER
OPEN EDITORS
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
OUTLINE
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
1: Code
Ln 77, Col 57 (142 selected) Spaces: 4 UTF-8 CRLF Python
Type here to search
8:53 PM
12/11/2020

b. Properly encrypt the data in transit between the client and the server.

- Before a message is sent from the client to the sever, the message itself is encrypted and then decrypted by the server once received.

3. For your choice of encryption:
   a. What method is being used for key exchange?

- RSA is used for key exchange.

   b. What method is being used for encryption?

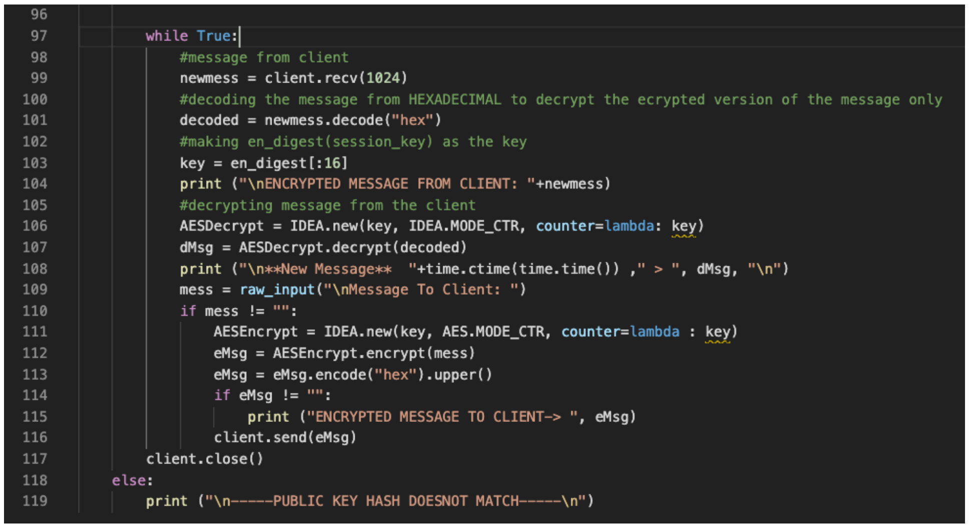
- AES is used for encryption.

   c. What message authentication code is being used?

- **The message authentication code being used is the CBC-MAC since the key is never used for anything else.**

4. Demonstrate that the data transferred between the client and server cannot be read by an eavesdropper with access to the network traffic.

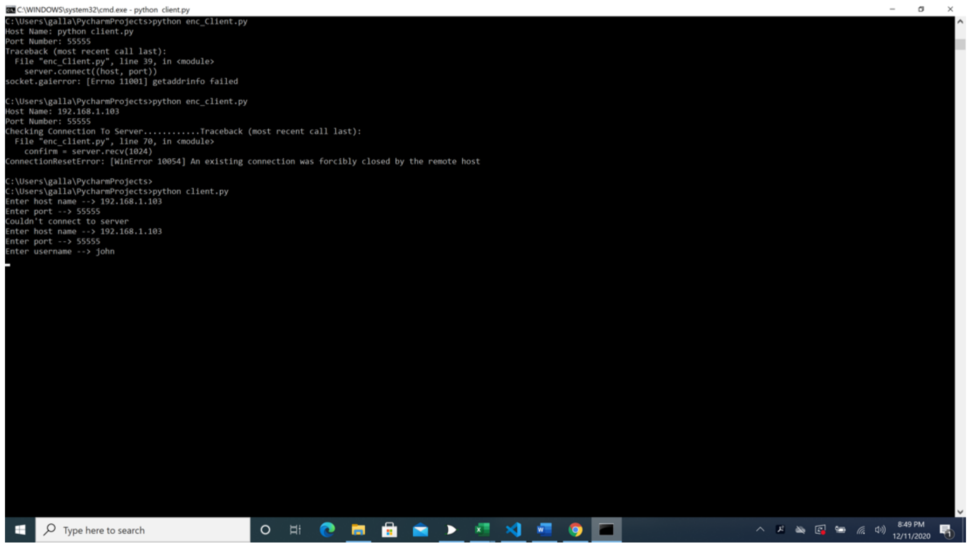
- As stated previously, an eavesdropper may not read the data transferred between the client and the server because the message is encrypted by the client before it is sent to the server. Below we demonstrate the process of the server having to decrypt the message when received from the client:



```python

        while True:
            #message from client
            newmess = client.recv(1024)
            #decoding the message from HEXADECIMAL to decrypt the ecrypted version of the message only
            decoded = newmess.decode("hex")
            #making en_digest(session_key) as the key
            key = en_digest[:16]
            print ("\nENCRYPTED MESSAGE FROM CLIENT: "+newmess)
            #decrypting message from the client
            AESDecrypt = IDEA.new(key, IDEA.MODE_CTR, counter=lambda: key)
            dMsg = AESDecrypt.decrypt(decoded)
            print ("\n**New Message**  "+time.ctime(time.time()) ," > ", dMsg, "\n")
            mess = raw_input("\nMessage To Client: ")
            if mess != "":
                AESEncrypt = IDEA.new(key, AES.MODE_CTR, counter=lambda : key)
                eMsg = AESEncrypt.encrypt(mess)
                eMsg = eMsg.encode("hex").upper()
                if eMsg != "":
                    print ("ENCRYPTED MESSAGE TO CLIENT-> ", eMsg)
                client.send(eMsg)
        client.close()
    else:
        print ("\n-----PUBLIC KEY HASH DOESNOT MATCH-----\n")
```

5. The project deliverables are:
    a. A report that answers all the questions provided, including the following:
        i. The results of running the original program.

- Below we have a screenshot of the original program once we ran it:



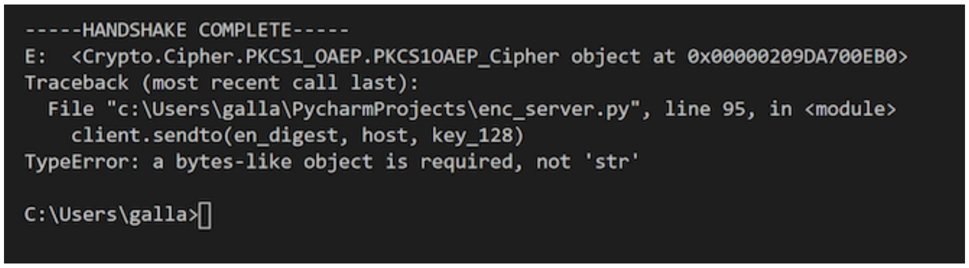
```
C:\WINDOWS\system32\cmd.exe - python  client.py
C:\Users\galla\PycharmProjects>python enc_Client.py
Host Name: python client.py
Port Number: 55555
Traceback (most recent call last):
  File "enc_Client.py", line 39, in <module>
    server.connect((host, port))
socket.gaierror: [Errno 11001] getaddrinfo failed

C:\Users\galla\PycharmProjects>python enc_client.py
Host Name: 192.168.1.103
Port Number: 55555
Checking Connection To Server............Traceback (most recent call last):
  File "enc_client.py", line 70, in <module>
    confirm = server.recv(1024)
ConnectionResetError: [WinError 10054] An existing connection was forcibly closed by the remote host

C:\Users\galla\PycharmProjects>
C:\Users\galla\PycharmProjects>python client.py
Enter host name --> 192.168.1.103
Enter port --> 55555
Couldn't connect to server
Enter host name --> 192.168.1.103
Enter port --> 55555
Enter username --> john
_
```

ii. Evidence from the demonstration that an eavesdropper can or cannot read the messages sent by the original program.

- As stated previously, the eavesdropper can read the messages sent by the original program since there is no encryption.

iii. An explanation of the mechanism that the client and server use to verify each other's identity.

- **Client side sends query strings when initializing the connection to the server. Once the server validates that token with the handshake, the server sets up the chat room. Below, we have a screenshot of the handshake being completed:**



```
-----HANDSHAKE COMPLETE-----
E:  <Crypto.Cipher.PKCS1_OAEP.PKCS1OAEP_Cipher object at 0x00000209DA700EB0>
Traceback (most recent call last):
  File "c:\Users\galla\PycharmProjects\enc_server.py", line 95, in <module>
    client.sendto(en_digest, host, key_128)
TypeError: a bytes-like object is required, not 'str'

C:\Users\galla>
```

iv. An explanation how the data is being encrypted.

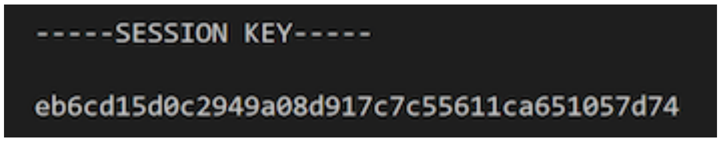
- **We used AES-CTR to encrypt and decrypt the messages.**

v. An explanation how key exchange is being handled.

- **The key exchange is being handled through hashing of the private and public key.**

vi. An explanation how authentication codes are being used.

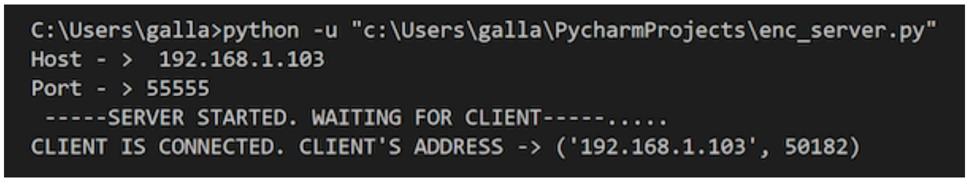
- **With every message sent, the same "session key" must be used as the authentication code. Below we have a screenshot of the specific session key for our program:**



```
-----SESSION KEY-----

eb6cd15d0c2949a08d917c7c55611ca651057d74
```

vii. Screenshots or the equivalent showing your code in action.

- **Below we have a screenshot of the code beginning the process:**



```
C:\Users\galla>python -u "c:\Users\galla\PycharmProjects\enc_server.py"
Host - >  192.168.1.103
Port - > 55555
 -----SERVER STARTED. WAITING FOR CLIENT-----.....
CLIENT IS CONNECTED. CLIENT'S ADDRESS -> ('192.168.1.103', 50182)
```

viii. Evidence from the demonstration that an eavesdropped cannot read the messages sent across the network.

- **An eavesdropper would not be able to read the messages sent across the network since the message is encrypted before it is sent from the client side and without the key the eavesdropper may not be able to decrypt it.**

b. The source code for your result.

i. It should be properly commented and written in a professional style.

ii. If the source code is contained in a publicly available repository (*e.g.,* GitHub) then it should have an appropriate license

- **We used a public GitHub repository to aid with the project. Below we have the MLA citation for it:**

- **The link:** https://gist.github.com/mjm918/b73d2d399d5b9c572cb7771f7ffd3b73
- **The citation:**

Julfikar, Mohammad. "Python Encryption Decryption (Socket Chat,RSA Encryption/Decryption,AES.MODE_CTR Encryption.IDEA.MODE_CTR Encryption/Decryption)." *Gist*, 21 Aug. 2020, gist.github.com/mjm918/b73d2d399d5b9c572cb7771f7ffd3b73.

iii. I will run the code, so any necessary instructions must be provided.

// Source Code Pasted Below

// Encrypted Server

```python
import socket
import hashlib
import os
import time
import itertools
import threading
import sys
import Crypto.Cipher.AES as AES
from Crypto.Util import Counter
from Crypto.Cipher import PKCS1_OAEP

from Crypto.PublicKey import RSA


#server address and port number input from admin
host= socket.gethostbyname(socket.gethostname())
print("Host - > ", host)
port = int(input("Port - > "))
#boolean for checking server and port
check = False
done = False

def animate():
    for c in itertools.cycle(['....','.......','..........','............']):
        if done:
            break
        sys.stdout.write('\rCHECKING IP ADDRESS AND NOT USED PORT '+ c)
        sys.stdout.flush()
        time.sleep(0.1)
    sys.stdout.write('\r -----SERVER STARTED. WAITING FOR CLIENT-----\n')
try:
    #setting up socket
    server = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    server.bind((host,port))
    server.listen(5)
    check = True
except BaseException:
    print("-----Check Server Address or Port-----")
    check = False

if check is True:
    # server Quit
    shutdown = False

    # printing "Server Started Message"
    thread_load = threading.Thread(target=animate)
```

```python
thread_load.start()

time.sleep(4)
done = True
#binding client and address
client,address = server.accept()
print ("CLIENT IS CONNECTED. CLIENT'S ADDRESS ->", address)
print ("\n-----WAITING FOR PUBLIC KEY & PUBLIC KEY HASH-----\n")

#client's message(Public Key)
getpbk = client.recv(2048)

#conversion of string to KEY
server_public_key = RSA.importKey(getpbk)

#hashing the public key in server side for validating the hash from client
hash_object = hashlib.sha1(getpbk)
hex_digest = hash_object.hexdigest()

if getpbk == "":
    print(getpbk)
    client.send("YES".encode())
    gethash = client.recv(1024)
    print("\n-----HASH OF PUBLIC KEY----- \n")
    print(gethash)
if hex_digest !=  gethash:
    # creating session key
    key_128 = os.urandom(16)
    #encrypt CTR MODE session key
    ctr = Counter.new(128, initial_value=0xff128eff)
    en = AES.new(key_128, AES.MODE_CTR, counter=ctr)

    #en = AES.new(os.urandom(32),AES.MODE_CTR,counter = lambda:key_128)
    encrypto = en.encrypt(key_128)
    #hashing sha1
    en_object = hashlib.sha1(encrypto)
    en_digest = en_object.hexdigest()

    print("\n-----SESSION KEY-----\n")
    print(en_digest)

    #encrypting session key and public key
    encryptor = PKCS1_OAEP.new(encrypto)
    E = encryptor

    print ("\n-----ENCRYPTED PUBLIC KEY AND SESSION KEY-----\n", str(E))
    print ("\n-----HANDSHAKE COMPLETE-----")
    print("E: ", E);
    client.sendto(en_digest.encode(),(address, 55555))

    while True:
            #message from client
            newmess = client.recv(1024)
            #decoding the message from HEXADECIMAL to decrypt the ecrypted versio
n of the message only
            decoded = newmess.decode("hex")
            #making en_digest(session_key) as the key
            key = en_digest[:16]
            print ("\nENCRYPTED MESSAGE FROM CLIENT: "+newmess)
            #decrypting message from the client
            AESDecrypt = IDEA.new(key, IDEA.MODE_CTR, counter=lambda: key)
            dMsg = AESDecrypt.decrypt(decoded)
            print ("\n**New Message**  "+time.ctime(time.time()) ," > ", dMsg, "\
n")
            mess = raw_input("\nMessage To Client: ")
            if mess != "":
                AESEncrypt = IDEA.new(key, AES.MODE_CTR, counter=lambda : key)
                eMsg = AESEncrypt.encrypt(mess)
                eMsg = eMsg.encode("hex").upper()
                if eMsg != "":
                    print ("ENCRYPTED MESSAGE TO CLIENT-> ", eMsg)
                client.send(eMsg)
        client.close()
    else:
        print ("\n-----PUBLIC KEY HASH DOESNOT MATCH-----\n")
```

// Encrypted Client

```
import time
import socket
import threading
import hashlib
import itertools
import sys
from Crypto import Random
from Crypto.Util import Counter
from Crypto.PublicKey import RSA


#loading menu
done = False
def menu():
    for c in itertools.cycle(['....','.......','..........','............']):
        if done:
            break
        sys.stdout.write('\rChecking Connection To Server'+c)
        sys.stdout.flush()
        time.sleep(0.1)

#Create Keys
random_generator = Random.new().read
key = RSA.generate(1024,random_generator)
public = key.publickey().exportKey()
private = key.exportKey()

#has keys
hash_object = hashlib.sha1(public)
hex_digest = hash_object.hexdigest()

#Set Socket
server = socket.socket(socket.AF_INET,socket.SOCK_STREAM)

#Select Host/Port
host = input("Host Name: ")
port = int(input("Port Number: "))
server.connect((host, port))
# print start message
thread_load = threading.Thread(target = menu)
thread_load.start()

time.sleep(4)
done = True
# AES CTR ENCRYPTION
def send(t, name, key):
    mess = raw_input(name + " : ")
    key = key[:16]
    # merege message to name
    whole = name + " : " + mess
    EncMessage =  IDEA.new(key, IDEA.MODE_CTR, counter=lambda : key)
    encMsg = EncMessage.encrypt(whole)
    #converting the encrypted message to HEXADECIMAL to readable
    encMsg = encMsg.encode("hex").upper()
    if encMsg != "":
        print ("ENCRYPTED MESSAGE TO HOST: "+encMsg)
    server.send(encMsg)
def recv(t,key):
    newmessage = server.recv(1024)
    print ("\nENCRYPTED MESSAGE FROM HOST: " + newmessage)
    key = key[:16]
    decoded = newmessage.decode("hex")
    AESDecrypt = IDEA.new(key, IDEA.MODE_CTR, counter=lambda: key)
    dMsg = AESDecrypt.decrypt(decoded)
    print ("\n**New Message From Server**  " + time.ctime(time.time()) + " : " +
dMsg + "\n")

while True:
    server.send(public)
    confirm = server.recv(1024)
    if confirm == "YES":
        server.send(hex_digest)

        #connected msg
        msg = server.recv(1024)
        enc = eval(msg)
        decrypt = key.decrypt(en)
        # hash
        enc_object = hashlib.sha1(decrypt)
        enc_digest = enc_object.hexdigest()

        print ("\n-----ENCRYPTED PUBLIC KEY AND SESSION KEY FROM SERVER-----")
        print (msg)
        print ("\n-----DECRYPTED SESSION KEY-----")
        print (en_digest)
        print ("\n-----HANDSHAKE COMPLETE-----\n")
        alais = input("\nYour Name -> ")

        while True:
            thread_send = threading.Thread(target = send , args = ( "-----
- Sending Message ------ ",alais,en_digest))
            thread_recv = threading.Thread(target=recv,args=( "------
Recieving Message------" ,en_digest))
            thread_send.start()
            thread_recv.start()

            thread_send.join()
            thread_recv.join()
            time.sleep(0.5)
        time.sleep(60)
        server.close()
```