

National Institute of Technology, Raipur

Department of Computer Science & Engineering



SECURE IP

A Term Project on Network Programming

GitHub Project Link: https://github.com/harsh9607/NP_Project.git

Submitted By:

Roll no: 14115036

Name: HARSH PATHAK

Abstract

Imagine a situation where you are talking to someone who is very important to you , close to you and all your chats become public , or some else who is not supposed to read it gets a hold of it. How would you feel ? Bitter..won't you ! Well...

For Human beings , they are emotional creatures and to them Privacy is of utmost importance. With half of the world being on social media , expressing feelings and emotions to each other behind the virtual screens its our responsibility as software engineers to make their communications secure and protect their Privacy ; thereby contributing to our Society and making it a better place. The project is all about secure communication.

SECURE IP is a project which will secure the communications by authenticating and encrypting each IP packet of a communication session; includes protocols for establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to be used during the session.

SECURE IP uses **RSA Algorithm** [2](Ronald L. Rivest, Adi Shamir and Leonard M. Adleman) for communicating between the sender and the receiver. Both the sender and receiver use public and private keys to encrypt and decrypt each others messages. Ronald L. Rivest, Adi Shamir and Leonard M. Adleman were awarded **Turing award** in the year 2002 for their contributions to Public Key Cryptography.

Initial authentication is done by **generating nonce** and by mutual exchange and verification of nonce we can actually authenticate if the other person is actually the designated receiver.

To ensure that the adversary can't tamper the messages and messages reach the receiver unhampered we use **Message digest** or **Hash Functions**. I have used **MD5** [3]/hash function in this project. Each message has a unique hash value which is verified at the receiver's end , every time a message is received. A detailed description of the project , its working and networking concepts will be discussed later in the project.

This was a brief overview of the project , lets hop onto the introduction now.

List of Figures

Figure 1: Socket Programming.....	5
Figure 2 Key Exchange	6
Figure 3 Alice generates Nonce	6
Figure 4 Bob Authenticates.....	6
Figure 5 Final Authentication.....	6
Figure 6 Flowchart	7
Figure 7 Encryption	8
Figure 8 Decryption & Verification	8
Figure 9 Screenshot 1.....	9
Figure 10 Screenshot 2.....	10

Table of Contents

Table of Contents

Abstract.....	2
List of Figures	3
Table of Contents.....	3
INTRODUCTION.....	4
MY MOTIVATION	4
PROBLEMS (Faced and overcame)	4
PROJECT DESCRIPTION	5
NETWORK PROGRAMING SIDE	5
CRYPTOGRAPHY SIDE	5
SCREENSHOTS (WORKING PROJECT)	9
Conclusion	11
References	12

INTRODUCTION

MY MOTIVATION

I love cryptography ; I did my internship in cryptography and now currently doing my minor project on cloud based encryption and so this topic was my obvious first choice. Moreover I am an ardent supporter of people's privacy and i believe that as software engineers this is my way of giving back to the society.

I use Python 2.7 because it is easy to use , quick, and my project uses libraries provided by python which are supported by both Linux and windows and hence its platform independent . Its not at all verbose like Java and comes with lot of functionalities and features.

PROBLEMS (Faced and overcame)

Although python makes your work pretty much easy and online community / forums like stack overflow are always there to ease of the pain , there were few road blocks that i had to overcome.

Firstly i had only read about public key Cryptography and never actually implemented it and hence it was practically a new domain. I knew that Python is widely used in Security and Ethical hacking and so python was my obvious choice. For the implemetation part during my internship I had discovered a library called Pycrpyto which helped me in implementing RSA algorithm. The greater challenge was exchanging of keys which was easily solved by using a built in function.

Second problem was how to authenticate that the message , ensuring it is not tampered by the attacker, and even if it was it should be easily detectable to the reciever and the conversation should stop. I solved this problem using hash function (MD5) that actually takes in you message and digests the entire message into a unique 128 bit hash value which looks completely random. Now even if the attacker changed the message by one bit, the entire hash would change completely thereby raising a flag that something is not right.

PROJECT DESCRIPTION

NETWORK PROGRAMING SIDE

```
# Creating a Socket Object , Type : TCP
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)

# To acquire the local machines IP address and binding it.
host = socket.gethostname()
s.bind((host,8888))

# server ready to listen
s.listen(1)

# Accepting connections
print('Server :: waiting for Connections:')
c , clientaddr = s.accept()
print('Server::Got a connection from client side')
```

Figure 1 : Socket programming[1]

- 1) A socket is created using the `socket.socket()` functions where `AF_INET` conotes the family and `socket.SOCK_STREAM` tells that it is a TCP connection
- 2) `gethostname()` acquires the machines IP address and `bind()` function binds the ip and port number with the file descriptor
- 3) `listen()` tells that the server is ready to accept connections and `accept()` function accepts the requests pending in the queue.

CRYPTOGRAPHY SIDE

```
#Public Key Exchanges
print('Server :: Sending my Public key')
c.send(publickey2send.exportKey());
ClientsPubKey = RSA.importKey(c.recv(2048))
print('Server :: Received Cilents Public key as well !')
print('Key exchange successful ! \n')
```

Figure 1: Socket Programming

KEY EXCHANGE

Here 'c' is the client side file descriptor (newsockfd) , we use built in function - c.send() to send our public key to the client side . Also the RSA.importKey() helps us to receive the clients public key.

Figure 2 Key Exchange

AUTHENTICATION

Lets say Alice and Bob want to talk to each other , but before the talk confidential information the want to authenticate each others identify.

```
nonce = str(random.randint(0,10000))
print('Nonce sent = ' + nonce)
E_nonce = ServersPubkey.encrypt(nonce,int(len(nonce)))
s.sendall(str(E_nonce))
```

Figure 3 Alice generates Nonce

<Alice's side> Nonce generation and sending it to Bob.

```
# Authentication Begins !!
temp = c.recv(2017)
d_nonce = key.decrypt(eval(temp))
print('nonce received ->' + d_nonce )
print('Authenticating !!!')

E_d_nonce = ClientsPubKey.encrypt(d_nonce,int(len(d_nonce)))
c.sendall(str(E_d_nonce))
```

Figure 4 Bob Authenticates

Figure 4 : <Bob's Side> Bob decrypts the nonce and sends the same nonce by encrypting it with Alices Public key

```
temp = s.recv(2017)
R_nonce = key.decrypt(eval(temp))

print(Figure 5 Final Authentication
if str(R_nonce) == str(nonce) :
    print('Authentication successful !!!')
else :
    print('Authentication failed')
    s.close()
```

FLOW CHART

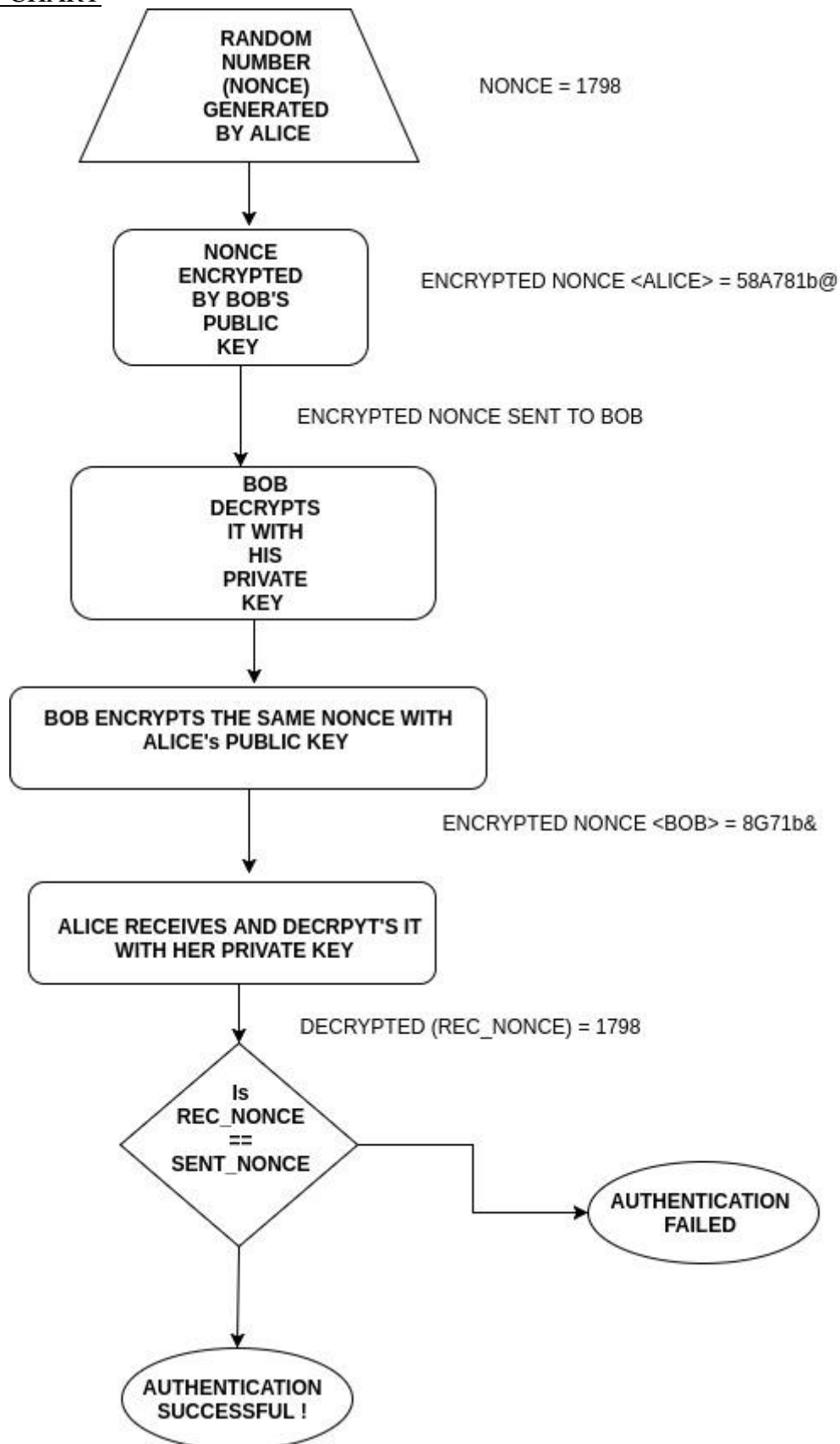


Figure 6 Flowchart

ENCRPTING MESSAGE

```
#ENCRYPTING YOUR MSG
Tencrypted = ServersPubkey.encrypt(msg,int(len(msg)))

s.sendall(str(Tencrypted))
s.recv(16) #dummy recv
s.send(md5_obj.hexdigest())

print('Your message in encrypted form looks like this ->' + str(Tencrypted))
```

Figure 7 Encryption

We use `PUBLIC_KEY.encrypt()` function to encrypt our message .
Params passed : message , length of the message.

DECRYPTION AND VERIFICATION

```
received_msg = s.recv(1024)
decrypted = key.decrypt(eval(received_msg))
md5_obj = hashlib.md5()
md5_obj.update(decrypted);

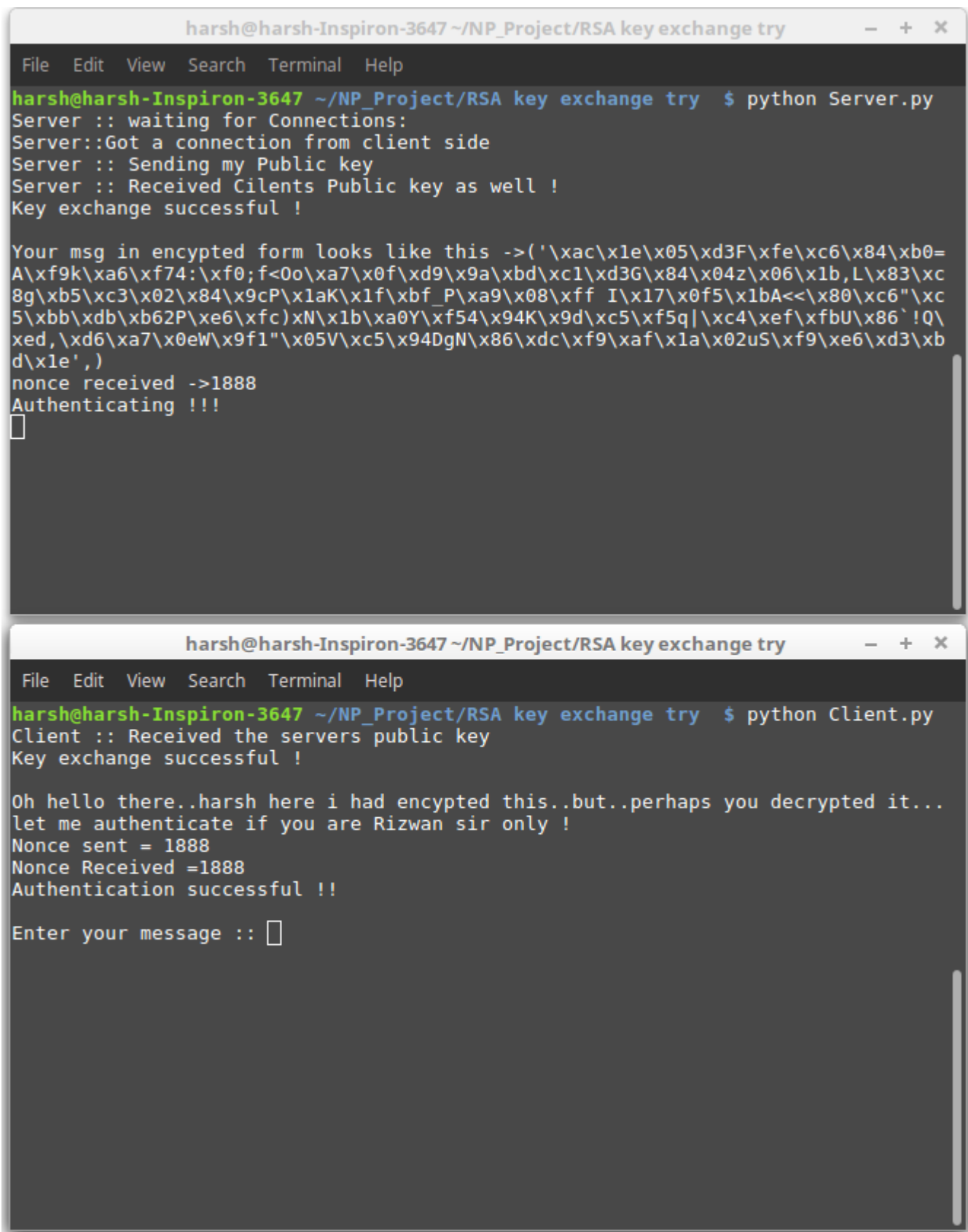
if md5_obj.hexdigest() != hashrecv :
    print('hashes dont match')
    break
```

Figure 8 Decryption & Verification

`PRIVATE_KEY.decrypt()` is a built in function to decrypt the received message.

As soon as the message is received , it gets stored in the our variable called `received_msg` ,
it is then decrypted , and converted into an MD5 object using `hashlib.md5()` function .
`MD5_obj.update()` commands helps to create the digest from the message.

SCREENSHOTS (WORKING PROJECT)



The image displays two screenshots of a terminal window titled "harsh@harsh-Inspiron-3647 ~/NP_Project/RSA key exchange try". The top screenshot shows the execution of "python Server.py". The server logs show it waiting for connections, receiving a connection from the client, sending its public key, and receiving the client's public key. It then displays a long, base64-encoded message and a nonce of 1888. The bottom screenshot shows the execution of "python Client.py". The client logs show it receiving the server's public key, sending a message (partially visible as "Oh hello there..harsh here i had encypted this..but..perhaps you decrypted it... let me authenticate if you are Rizwan sir only !"), and receiving the server's nonce. Both sides confirm successful authentication.

```
harsh@harsh-Inspiron-3647 ~/NP_Project/RSA key exchange try $ python Server.py
Server :: waiting for Connections:
Server::Got a connection from client side
Server :: Sending my Public key
Server :: Received Cilents Public key as well !
Key exchange successful !

Your msg in encrypted form looks like this ->('\xac\x1e\x05\xd3F\xfe\xc6\x84\xb0=
A\x9f9k\xa6\xf74:\xf0;f<0o\xa7\x0f\xd9\x9a\xbd\xc1\xd3G\x84\x04z\x06\x1b,L\x83\xc
8g\xb5\xc3\x02\x84\x9cP\x1aK\x1f\xbf_P\xa9\x08\xff I\x17\x0f5\x1bA<<\x80\xc6"\xc
5\xbb\xdb\xb62P\xe6\xfc)xN\x1b\xa0Y\xf54\x94K\x9d\xc5\xf5q|\xc4\xef\xfbU\x86`!Q\
xed,\xd6\xa7\x0eW\x9f1"\x05V\xc5\x94DgN\x86\xdc\xf9\xaf\x1a\x02u5\xf9\xe6\xd3\xb
d\x1e',)
nonce received ->1888
Authenticating !!!
█

harsh@harsh-Inspiron-3647 ~/NP_Project/RSA key exchange try $ python Client.py
Client :: Received the servers public key
Key exchange successful !

Oh hello there..harsh here i had encypted this..but..perhaps you decrypted it...
let me authenticate if you are Rizwan sir only !
Nonce sent = 1888
Nonce Received =1888
Authentication successful !!

Enter your message :: █
```

Figure 9 Screenshot 1

```
harsh@harsh-Inspiron-3647 ~/NP_Project/RSA key exchange try - + x
File Edit View Search Terminal Help
harsh@harsh-Inspiron-3647 ~/NP_Project/RSA key exchange try $ python Server.py
Server :: waiting for Connections:
Server::Got a connection from client side
Server :: Sending my Public key
Server :: Received Cilents Public key as well !
Key exchange successful !

Your msg in encrypted form looks like this ->('\xac\x1e\x05\xd3F\xfe\xc6\x84\xb0=
A\x9f9k\xa6\xf74:\xf0;f<0o\xa7\x0f\xd9\x9a\xbd\xc1\xd3G\x84\x04z\x06\x1b,L\x83\xc
8g\xb5\xc3\x02\x84\x9cP\x1aK\x1f\xbf_P\xa9\x08\xff I\x17\x0f5\x1bA<<\x80\xc6"\xc
5\xbb\xdb\xb62P\xe6\xfc)xN\x1b\xa0Y\xf54\x94K\x9d\xc5\xf5q|\xc4\xef\xfbU\x86`!Q\
xed,\xd6\xa7\x0eW\x9f1"\x05V\xc5\x94DgN\x86\xdc\xf9\xaf\x1a\x02uS\xf9\xe6\xd3\xb
d\x1e',)
nonce received ->1888
Authenticating !!!
█

harsh@harsh-Inspiron-3647 ~/NP_Project/RSA key exchange try - + x
File Edit View Search Terminal Help
harsh@harsh-Inspiron-3647 ~/NP_Project/RSA key exchange try $ python Client.py
Client :: Received the servers public key
Key exchange successful !

Oh hello there..harsh here i had encrypted this..but..perhaps you decrypted it...
let me authenticate if you are Rizwan sir only !
Nonce sent = 1888
Nonce Received =1888
Authentication successful !!

Enter your message :: █
```

Figure 10 Screenshot 2

Conclusion

I would like to conclude by saying that SECURE IP is a full fledged project that allows two users to communicate in a secure fashion . State of the Art RSA Algorithm has been used in the project. Message authentication & verification helps to preserve the privacy of the message . I wanted to make an actual practically usable project , that can be used by people. RSA provides you protection from Bruteforce attacks, ciphertext only attack etc.

From my side i have tried my level best to explain the project in the most lucid and simple manner by adding as many images as possible.

I would conclude by thanking my Professor Mr Rizwan Rawani for believing in me like he does in all the students.

Thanking you ..

Harsh Pathak
7th Semester
14115036
B.Tech

References

[1]Tutorials Point . 2017.Socket Programming retrived from https://www.tutorialspoint.com/python3/python_networking.htm

[2]Python Software foundation(US). Pycrypto 2.6.1. RSA Algorithm retrived from <https://pypi.python.org/pypi/pycrypto>

[3]Python Software foundation(US). 14.1 hashlib .Message digest algorithm MD5 retrived from <https://docs.python.org/2/library/hashlib.html#module-hashlib>