ASSIGNMENT - 6

C_RSA.py

```
# Import socket module
import socket
from random import randint
import math

def generate_prime():# generate random prime function
	x = randint(1, 25)
	while True:
		if is_prime(x):
			break
		else:
			x += 1
	return x


def is_prime(x):# primality check function
	i = 2
	root = math.ceil(math.sqrt(x))
	while i <= root:
		if x % i == 0:
			return False
		i += 1
	return True

# function to find gcd
def gcd(a, b):
	while b:
		a, b = b, a%b
	return a

# function to find extended gcd
def egcd(a, b):
	if a == 0:
		return (b, 0, 1)
	else:
		g, y, x = egcd(b % a, a)
		return (g, x - (b // a) * y, y)

# function to find modular inverse
def modinv(a,m):
	g,x,y = egcd(a,m)
	if g != 1:
		return None
```

```
        else:
                return x%m
# next create a socket object

pb = 13
qb = 17
nb = pb * qb

nb1 = (pb - 1) * (qb - 1)

r = randint(2,100) # For efficiency 2 < e < 100
while True:
        if gcd(r, nb1) == 1:
                break
        else:
                r += 1
eb = r
#print("ea = %d" % eb)

        # Compute d, the modular multiplicative inverse of e
        # Private key exponent d
private_b = modinv(eb, nb1)
print("Private key of B is: %d" % private_b)


public_b = str(str(eb)+" " +str(nb) )
print("Public key of B is: " + public_b)
# Create a socket object
s = socket.socket()

# Define the port on which you want to connect
port = 1232

# connect to the server on local computer
s.connect(('127.0.0.1', port))

# receive data from the server and decoding to get the string.

s.sendall(public_b.encode())
info_a = s.recv(1024).decode().split()
ea = int(info_a[0])
na = int(info_a[1])
#print (ea)
#print (na)

message_b = int(input('Enter message for server: '))
encrp_msg_b = str((message_b**private_b) % nb)

s.sendall(encrp_msg_b.encode())
```

```
message_a = int(s.recv(1024).decode())
print('Encrpyted message from server recieved: %d' %message_a)
final_msg = (int(message_a)**ea) % na
print('After decryption: %d' %final_msg)

# close the connection
s.close()
```

S_RSA.py

```
#first of all import the socket library
import socket
from random import randint
import math

def generate_prime():# generate random prime function
	x = randint(1, 15)
	while True:
		if is_prime(x):
			break
		else:
			x += 1
	return x


def is_prime(x):# primality check function
	i = 2
	root = math.ceil(math.sqrt(x))
	while i <= root:
		if x % i == 0:
			return False
		i += 1
	return True

# function to find gcd
def gcd(a, b):
	while b:
		a, b = b, a%b
	return a

# function to find extended gcd
def egcd(a, b):
	if a == 0:
		return (b, 0, 1)
	else:
```

```
                g, y, x = egcd(b % a, a)
                return (g, x - (b // a) * y, y)

# function to find modular inverse
def modinv(a,m):
        g,x,y = egcd(a,m)
        if g != 1:
                return None
        else:
                return x%m
# next create a socket object

pa = 13
qa = 17
na = pa * qa

na1 = (pa - 1) * (qa - 1)

r = randint(2,100) # For efficiency 2 < e < 100
while True:
        if gcd(r, na1) == 1:
                break
        else:
                r += 1
ea = r
#print("ea = %d" % ea)

        # Compute d, the modular multiplicative inverse of e
        # Private key exponent d
private_a = modinv(ea, na1)
print("Private key of A is: = %d" % private_a)

public_a = str(str(ea) + " " + str(na) )
print("Public key of A is: " + public_a)
s = socket.socket()
print ("Socket successfully created")

# reserve a port on your computer in our
# case it is 12345 but it can be anything
port = 1232

# Next bind to the port
# we have not typed any ip in the ip field
# instead we have inputted an empty string
# this makes the server listen to requests
# coming from other computers on the network
s.bind(('', port))
print ("socket binded to %s" %(port))
```

```
# put the socket into listening mode
s.listen(5)
print ("socket is listening")

# a forever loop until we interrupt it or
# an error occurs
while True:

# Establish connection with client.
  c, addr = s.accept()
  print ('Got connection from', addr )

  # send a thank you message to the client. encoding to send byte type.
  c.send(public_a.encode())

  info_b = c.recv(1024).decode().split(" ")
  # print(info_b)
  eb = int(info_b[0])
  nb = int(info_b[1])
  #print (eb)
  #print (nb)

  message_a = int(input('Enter message for client: '))
  encrp_msg_a = str((message_a**private_a) % na)

  c.send(encrp_msg_a.encode())


  message_b = int(c.recv(1024).decode())
  print('Encrpyted message from client recieved: %d' %message_b)
  final_msg = (int(message_b)**eb) % nb
  print('After decryption: %d' %final_msg)
  # Close the connection with the client
  c.close()

  # Breaking once connection closed
  break
```

OUTPUT

SERVER OUTPUT

Private key of A is: 37
Public key of A is: 17 221
Socket successfully created
socket binded to 1232
socket is listening
Got connection from ('127.0.0.1', 56789)
Enter message for client: 42
Encrypted message from client received: 98
After decryption: 99

CLIENT OUTPUT

Private key of B is: 53
Public key of B is: 23 221
Enter message for server: 99
Encrypted message from server received: 176
After decryption: 42