Name: Shrikant Pawar
Roll No.: 57
Assignment No. 07

Title: Implement a client and a server on different computers using python. Perform the communication between these two entities by using RSA cryptosystem.

Server Code:

```
import socket
import time
import string
from diffie_hellman import getLargePrimeNumber, getPrimitiveRoot, keyGeneration,
sharedKeyGeneration
from des import DES_Algorithm

serverPort = 8001
serverIP = "127.0.0.1"


def keyGenerationForDES(p, q, sharedKey):
    '''
    This is just a function to generate a key of sufficient length
    for the DES Algorithm to work using the shared key formed and the
    global parameters
    '''
    mapping = {}
    for index, letter in enumerate(string.ascii_letters):
        mapping[index] = letter

    val = str(sharedKey * p * q)

    finalKey = []
    for index in range(0, len(val), 2):
        finalKey.append(mapping[int(val[index:index + 1]) % len(mapping)])

    while len(finalKey) < 8:
        finalKey += finalKey

    return "".join(finalKey[:8])


def main():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((serverIP, serverPort))
    server.listen(1)  # max backlog of connections

    # Establishing the connection
    print("Establishing connection with client")
    client_sock, address = server.accept()
    print(client_sock.recv(4096).decode())

    # Setting the Global Parameters
    p = getLargePrimeNumber(1000, 2000)
    q = getPrimitiveRoot(p, True)

    print("Forwarding Global Parameters to Client")
    client_sock.send(str(p).encode())
    # Time lag needed else p & q concatenate for some reason
    time.sleep(2)
    client_sock.send(str(q).encode())

    # Generating the Public-Private Key Pair
    privateServer, publicServer = keyGeneration(p, q)
    time.sleep(2)

    # Sending the Server Public Key
    client_sock.send(str(publicServer).encode())

    # Receiving the Public Key from Client
    publicClient = int(client_sock.recv(4096).decode())

    time.sleep(2)

    key = int(str(sharedKeyGeneration(publicClient, privateServer, p)), 16)
    DES_key = keyGenerationForDES(p, q, key)

    while True:
        actual_message = client_sock.recv(4096).decode()
        message = DES_Algorithm(text=actual_message,
                                key=DES_key, encrypt=False).DES()
        if message != "exit":
            print("Peer says: " + message)
            print("The message recieved: {0}".format(actual_message))
            message_to_send = input("You: ")
            encrytedMessage = DES_Algorithm(
                text=message_to_send, key=DES_key, encrypt=True).DES()
            client_sock.send(encrytedMessage.encode())
        else:
            client_sock.close()


if __name__ == '__main__':
    main()
```