Name: **Mareena Fernandes**
Roll No.: **8669**
Class: **TE IT**
Batch: **B**

EXPERIMENT NO: 3

Client:

```
"""
Need server initialized first binded to CONN_PORT and listening or will crash
"""

import socket


def power_mod(base, power, mod):
    res = 1
    while power > 0:
        if power % 2 == 1:
            res = (res * base) % mod
        power = power // 2
        base = (base * base) % mod
    return res


sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
CONN_ADDR = socket.gethostname()

# Use for regular diffie hellman
CONN_PORT = 1234

# Use for man in the middle diffie hellman
# CONN_PORT = 1235

try:
    print("Attempting connection to server...")
    sock.connect((CONN_ADDR, CONN_PORT))

    public_key_str = sock.recv(1024).decode("utf-8")
```

```
    public_key = [int(x) for x in public_key_str.split()]
    print("Server Public Key: ", public_key)

    secret_client = int(input("Enter your secret key: "))
    half_key_client = power_mod(public_key[0], secret_client, public_key[1])
    sock.send(str(half_key_client).encode())

    half_key_server = int(sock.recv(1024).decode("utf-8"))
    print("Server half key: ", half_key_server)

    shared_key = power_mod(half_key_server, secret_client, public_key[1])
    print("Shared key: ", shared_key)

except:
    print("Something went wrong")

finally:
    print("Closing connection to server")
    sock.close()
```

Server:

```
import socket


def power_mod(base, power, mod):
    res = 1
    while power > 0:
        if power % 2 == 1:
            res = (res * base) % mod
        power = power // 2
        base = (base * base) % mod
    return res


public_key_str = input("Enter 2 random integers for public key: ")
public_key = [int(x) for x in public_key_str.split()]
secret_server = int(input("Enter server secret key: "))

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
SERV_ADDR = socket.gethostname()
SERV_PORT = 1234
sock.bind((SERV_ADDR, SERV_PORT))

# listen parameter holds max unaccepted connections at a time
sock.listen(3)
print("Waiting for client connection...")

try:
    client, caddr = sock.accept()
    print("Connected to ", caddr)
    client.send(public_key_str.encode())

    half_key_client = int(client.recv(1024).decode("utf-8"))
    print("Client half key: ", half_key_client)

    half_key_server = power_mod(public_key[0], secret_server, public_key[1])
    client.send(str(half_key_server).encode())
```

```
    shared_key = power_mod(half_key_client, secret_server, public_key[1])
    print("Shared Key: ", shared_key)

except:
    print("Something went wrong")

finally:
    print("Closing connection to client")
    client.close()
```

Post labs:

Demonstrate Man in Middle Attack by creating a new process that intrudes in between

```
"""
For man in the middle the server and mim must be initialized first. Client port must be set to mim po
rt and then client initialization. Server will finish first when mim sends half key to server. client and
mim will finish last sending each other half keys after receiving public key from server.
"""

import socket


def power_mod(base, power, mod):
    res = 1
    while power > 0:
        if power % 2 == 1:
            res = (res * base) % mod
        power = power // 2
        base = (base * base) % mod
    return res


SERV_ADDR = socket.gethostname()
SERV_PORT = 1234

ATT_ADDR = SERV_ADDR
ATT_PORT = 1235

secret_att = int(input("Enter attacker secret: "))

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind((ATT_ADDR, ATT_PORT))
sock.listen(3)
print("Waiting for Client (user 1) to open connection...")

try:
    client, caddr = sock.accept()
    print("Connected to Client (user 1) at", caddr)
```

```
    print("Attempting connection to Server (user 2)...")
    serv = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    serv.connect((SERV_ADDR, SERV_PORT))

    public_key_str = serv.recv(1024).decode("utf-8")
    client.send(public_key_str.encode())

    public_key = [int(x) for x in public_key_str.split()]
    print("Server public key: ", public_key)

    attacker_half_key = power_mod(public_key[0], secret_att, public_key[1])
    serv.send(str(attacker_half_key).encode())
    half_key_2 = int(serv.recv(1024).decode("utf-8"))
    print("Server half key:", half_key_2)

    half_key_1 = int(client.recv(1024).decode("utf-8"))
    print("Client half key: ", half_key_1)
    client.send(str(attacker_half_key).encode())

    shared_key_att_1 = power_mod(half_key_1, secret_att, public_key[1])
    shared_key_att_2 = power_mod(half_key_2, secret_att, public_key[1])
    print("Client (user 1) and attacker shared key: ", shared_key_att_1)
    print("Server (user 2) and attacker shared key: ", shared_key_att_2)

except:
    print("Something went wrong")

finally:
    print("Closing both connections")
    serv.close()
    client.close()
```