3. Implement a client and a server on different computers using python. Perform the encryption of

message ofsender between these two entities by using DES Algorithm and use Diffie Hellman method

for exchange of keys.

Server Code:

import socket

import random

from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes

from cryptography.hazmat.primitives import padding

from cryptography.hazmat.primitives.asymmetric import dh, utils

from cryptography.hazmat.primitives.serialization import Encoding, PublicFormat

# Generate parameters for Diffie-Hellman key exchange

parameters = dh.generate\_parameters(generator=2, key\_size=512)

private\_key = parameters.generate\_private\_key()

public\_key = private\_key.public\_key().public\_bytes(Encoding.PEM, PublicFormat.SubjectPublicKeyInfo)

# Create socket and bind to a specific port

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_address = ('localhost', 12345)

sock.bind(server\_address)

# Listen for incoming connections

sock.listen(1)

while True:

print('Waiting for a connection...')

connection, client\_address = sock.accept()

try:

print('Connection from', client\_address)

# Send public key to client

connection.sendall(public\_key)

# Receive client's public key for Diffie-Hellman key exchange

client\_public\_key\_bytes = connection.recv(1024)

client\_public\_key = dh.DHPublicKey.from\_encoded\_point(parameters, client\_public\_key\_bytes)

# Perform Diffie-Hellman key exchange to derive shared secret

shared\_secret = private\_key.exchange(client\_public\_key)

# Generate key and IV for DES encryption

key = shared\_secret[:8]

iv = shared\_secret[8:16]

# Receive message from client and decrypt it using DES

ciphertext = connection.recv(1024)

cipher = Cipher(algorithms.DES(key), modes.CBC(iv))

decryptor = cipher.decryptor()

unpadder = padding.PKCS7(64).unpadder()

decrypted\_message = unpadder.update(decryptor.update(ciphertext)) + unpadder.finalize()

print(f"Received message from client: {decrypted\_message.decode('utf-8')}")

finally:

# Close the connection

connection.close()

Client Code :

import socket

import random

from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes

from cryptography.hazmat.primitives import padding

from cryptography.hazmat.primitives.asymmetric import dh, utils

from cryptography.hazmat.primitives.serialization import Encoding, PublicFormat

# Create socket and connect to server

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_address = ('localhost', 12345)

sock.connect(server\_address)

try:

# Generate parameters for Diffie-Hellman key exchange

parameters = dh.generate\_parameters(generator=2, key\_size=512)

private\_key = parameters.generate\_private\_key()

public\_key = private\_key.public\_key().public\_bytes(Encoding.PEM, PublicFormat.SubjectPublicKeyInfo)

# Send client's public key for Diffie-Hellman key exchange

sock.sendall(public\_key)

# Receive server's public key for Diffie-Hellman key exchange

server\_public\_key\_bytes = sock.recv(1024)

server\_public\_key = dh.DHPublicKey.from\_encoded\_point(parameters, server\_public\_key\_bytes)

# Perform Diffie-Hellman key exchange to derive shared secret

shared\_secret = private\_key.exchange(server\_public\_key)

# Generate key and IV for DES encryption

key = shared\_secret[:8]

iv = shared\_secret[8:16]

# Encrypt message using DES and send it

To implement a client-server communication with DES encryption using Diffie-Hellman key exchange in Python, we can use the **socket** and **cryptography** modules. Here's an example implementation: