**Key Exchange using Diffie-Hellman**

**Server**

import socket

import random

# Diffie-Hellman key exchange

def generate\_prime():

prime = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]

u\_n = random.sample(prime, len(prime))

p, q = u\_n[1], u\_n[2]

return p,q

def generate\_key\_pair(p, g):

#private\_key = random.randint(2, p - 1)

private\_key =int(input("enter key: "))

public\_key = pow(g, private\_key, p)

return private\_key, public\_key

def calculate\_shared\_secret(private\_key, public\_key, p):

return pow(public\_key, private\_key, p)

# Set up the server

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.bind(('192.168.1.24', 8080))

server\_socket.listen(1)

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

# Accept a connection

client\_socket, client\_address = server\_socket.accept()

print(f"Connection from {client\_address}")

p,g= generate\_prime()

# Send primes to the client

primes\_message = f"{p},{g}"

client\_socket.send(primes\_message.encode())

bob\_private\_key, bob\_public\_key = generate\_key\_pair(p, g)

print("Public key of b: ",bob\_public\_key)

client\_socket.send(str(bob\_public\_key).encode())

client\_public\_key = int(client\_socket.recv(1024).decode())

print("Public key of A: ",client\_public\_key)

shared\_secret\_bob = calculate\_shared\_secret(bob\_private\_key, client\_public\_key, p)

print("Shared secret key with A:", shared\_secret\_bob)

# Close the connection

client\_socket.close()

server\_socket.close()

**Client**

import socket

import random

def generate\_key\_pair(p, g):

#private\_key = random.randint(2, p - 1)

private\_key = int(input("Enter Private key"))

public\_key = pow(g, private\_key, p)

return private\_key, public\_key

def calculate\_shared\_secret(private\_key, public\_key, p):

return pow(public\_key, private\_key, p)

# Set up the client

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client\_socket.connect(('192.168.1.47', 8080))

# Diffie-Hellman key exchange

primes\_message = client\_socket.recv(1024).decode()

p, g = map(int, primes\_message.split(','))

print(f"Prime no.: {p} \n value of g: {g}")

alice\_private\_key, alice\_public\_key = generate\_key\_pair(p, g)

print("Public key of A: ",alice\_public\_key)

client\_socket.send(str(alice\_public\_key).encode())

server\_public\_key = int(client\_socket.recv(1024).decode())

print("Public key of B: ",server\_public\_key)

shared\_secret\_alice = calculate\_shared\_secret(alice\_private\_key, server\_public\_key, p)

print("Shared key with B:", shared\_secret\_alice)

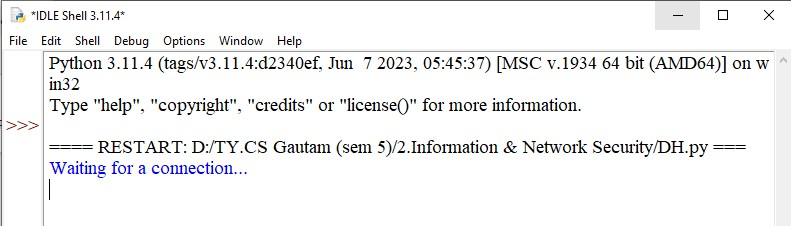
# Close the connection

client\_socket.close()

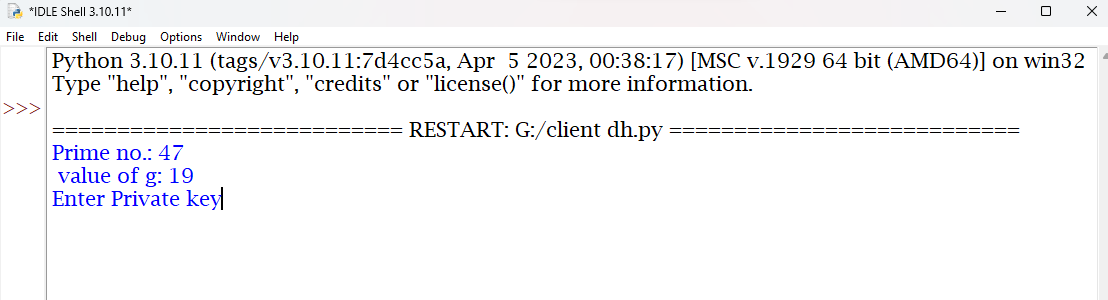
**Output**

**First execute the server program**

**Server wating for connection**

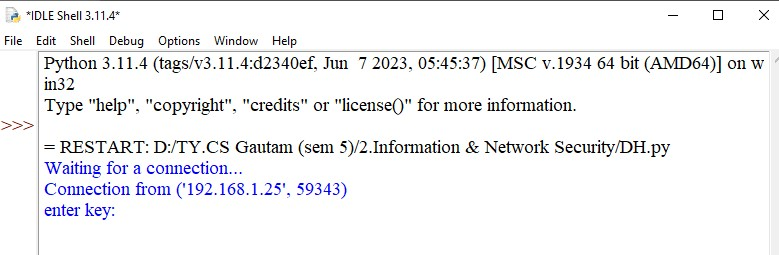


**Then execute the client program**

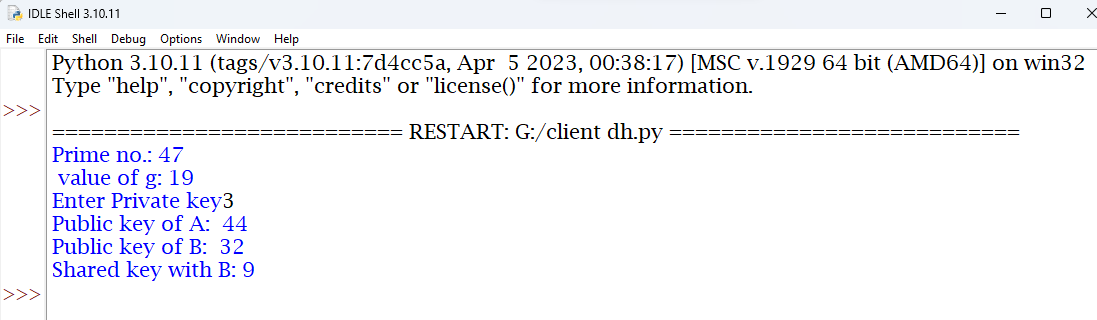
****

**Server side Output**

**Connection successfully**



**Client enter the Private key then server enter the private key**

****

**Output at server side**

