PART A

(PART A: TO BE REFERRED BY STUDENTS)

Experiment No.06

A.1—Aim:

Demonstrate the used of Diffie-Hellman Key Exchange algorithm and Implementation of Diffie-Hellman algorithm.

A.2--- Prerequisite:

1. Fundamentals of Symmetric cryptography.

A.3--- Outcome:

After completing this experiment, you will be able to:

- To Learn about working of Key Exchange Algorithm.
- To Understand the Diffie-Hellman algorithm.
- To Implement the Diffie Hellman algorithm.

A.4--- Theory:

- The Diffie-Hellman key exchange works by allowing two parties (Alice and Bob) to agree on a shared secret key over an insecure channel, without any other party being able to intercept the key or learn anything about it.
- The Diffie-Hellman key exchange (also known as exponential key exchange) is a method for securely exchanging cryptographic keys over an insecure channel.
- It is a fundamental building block of many secure communication protocols, including SSL/TLS and SSH.

(PART - B)

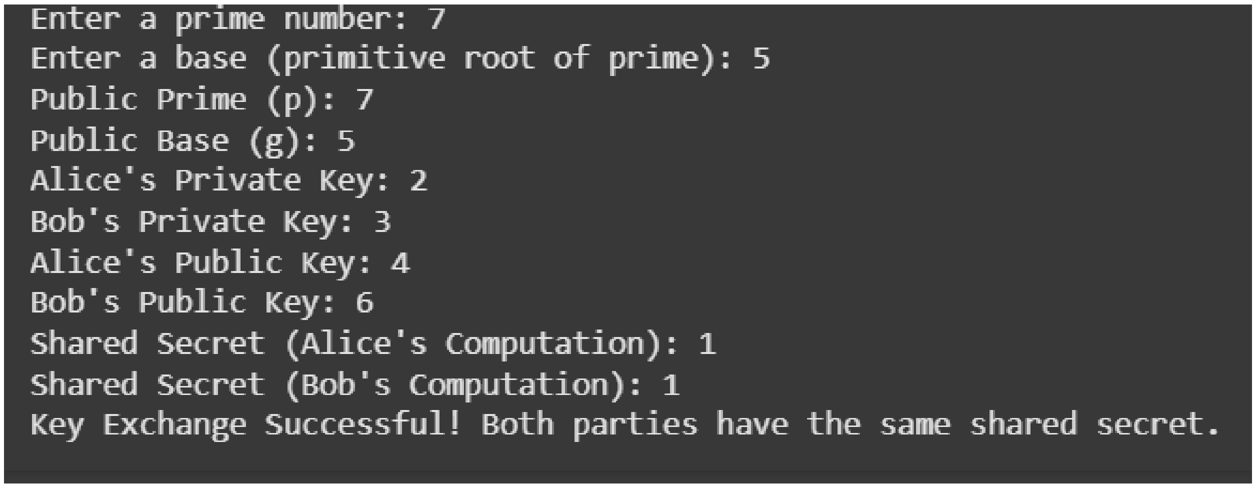
(TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per the following segments within two hours of the practical. The soft copy must be submitted on the portal or on MS Teams before the deadline)

Roll.No. : C175	Name: Nidhish Rathod
Sem/Year : X/5th	Batch: D2
Date of Experiment :	Date of Submission:
Grade --	

B.1: Procedure of performed experiment

```
import random

def generate_private_key(prime):
    return random.randint(2, prime - 2)

def compute_public_key(base, private_key, prime):
    return pow(base, private_key, prime)

def compute_shared_secret(public_key, private_key, prime):
    return pow(public_key, private_key, prime)

def diffie_hellman():

    prime = int(input("Enter a prime number: "))
    base = int(input("Enter a base (primitive root of prime): "))

    private_key_alice = generate_private_key(prime)
    private_key_bob = generate_private_key(prime)

    public_key_alice = compute_public_key(base, private_key_alice, prime)
    public_key_bob = compute_public_key(base, private_key_bob, prime)

    shared_secret_alice = compute_shared_secret(public_key_bob, private_key_alice, prime)
    shared_secret_bob = compute_shared_secret(public_key_alice, private_key_bob, prime)

    print(f"Public Prime (p): {prime}")
    print(f"Public Base (g): {base}")
    print(f"Alice's Private Key: {private_key_alice}")
    print(f"Bob's Private Key: {private_key_bob}")
    print(f"Alice's Public Key: {public_key_alice}")
    print(f"Bob's Public Key: {public_key_bob}")
    print(f"Shared Secret (Alice's Computation): {shared_secret_alice}")
    print(f"Shared Secret (Bob's Computation): {shared_secret_bob}")

    if shared_secret_alice == shared_secret_bob:
        print("Key Exchange Successful! Both parties have the same shared secret.")
    else:
        print("Key Exchange Failed! Shared secrets do not match.")

diffie_hellman()
```



```
Enter a prime number: 7
Enter a base (primitive root of prime): 5
Public Prime (p): 7
Public Base (g): 5
Alice's Private Key: 2
Bob's Private Key: 3
Alice's Public Key: 4
Bob's Public Key: 6
Shared Secret (Alice's Computation): 1
Shared Secret (Bob's Computation): 1
Key Exchange Successful! Both parties have the same shared secret.
```

Example:

<Write any example of Diffie Hellman Algorithm (numerical problem)>

Demonstrate MITM attack working on Diffie-Hellman:

Let's assume Alice and Bob agree on:
A prime number (p) = 23
A base (g) = 5
Now, they each select a private key:
Alice chooses private key = 6
Bob chooses private key = 15
They compute their public keys:
Alice: PA=gprivateAmod p=56mod 23=8P_A = g^{privateA} \mod p = 5^6 \mod 23 = 8PA =gprivateAmodp=56mod23=8
Bob: PB=gprivateBmod p=515mod 23=19P_B = g^{privateB} \mod p = 5^{15} \mod 23 = 19PB =gprivateBmodp=515mod23=19
They exchange public keys and compute the shared secret:
Alice: SA=PBprivateAmod p=196mod 23=2S_A = P_B^{privateA} \mod p = 19^6 \mod 23 = 2SA =PBprivateAmodp=196mod23=2
Bob: SB=PAprivateBmod p=815mod 23=2S_B = P_A^{privateB} \mod p = 8^{15} \mod 23 = 2SB =PAprivateBmodp=815mod23=2
Since SA=SBS_A = S_BSA=SB, Alice and Bob now have a shared secret key 2.

- Man-in-the-Middle (MITM) Attack on Diffie-Hellman

Let's assume a malicious attacker, Mallory, intercepts the exchange between Alice and Bob.
Alice sends her public key PA=8P_A = 8PA=8 to Bob, but Mallory intercepts it.
Mallory generates her own private key K_M = 10, computes PM=gKMmod p=510mod 23=9P_M = g^{K_M} \mod p = 5^{10} \mod 23 = 9PM=gKMmodp=510mod23=9, and sends it to Bob instead of Alice's real public key.
Similarly, when Bob sends his public key PB=19P_B = 19PB=19, Mallory intercepts it and replaces it with her own PM=9P_M = 9PM=9, sending it to Alice.
Now, Alice computes the shared secret using PMP_MPM, and so does Bob, but both actually share a secret with Mallory instead of each other:
Alice: SA=96mod 23=4S_A = 9^6 \mod 23 = 4SA=96mod23=4 (secret with Mallory)
Bob: SB=915mod 23=16S_B = 9^{15} \mod 23 = 16SB=915mod23=16 (secret with Mallory)
Mallory computes:
SA=810mod 23=4S_A = 8^{10} \mod 23 = 4SA=810mod23=4 (secret with Alice)
SB=1910mod 23=16S_B = 19^{10} \mod 23 = 16SB=1910mod23=16 (secret with Bob)
Now, Mallory has successfully established two separate secrets with Alice and Bob, allowing her to intercept and modify messages without them knowing.

B.2: Observations and Learning’s:

The Diffie-Hellman Key Exchange enables secure key sharing over an insecure channel but is vulnerable to Man-in-the-Middle (MITM) attacks without authentication. Implementing authentication mechanisms like digital signatures or PKI enhances its security.

B.3: Conclusion:

The Diffie-Hellman Key Exchange is a secure method for establishing a shared secret over an insecure channel, relying on the difficulty of solving discrete logarithms. However, it is vulnerable to Man-in-the-Middle (MITM) attacks without authentication, which can be mitigated using digital signatures or PKI.