Terna Engineering College

Computer Engineering Department

Class: TE **Sem.: VI**

Course: System Security Lab

PART A

(PART A : TO BE REFFERED BY STUDENTS)

Experiment No.03

A.1 Aim:

Implementation of Diffie Hellman Key exchange algorithm.

A.2 Prerequisite:

1. Understanding of modular arithmetic.
2. Basic knowledge of Asymmetric Key Cryptography and secure communication.

A.3 Outcome:

After the successful completion of this experiment, students will be able to:

1. Analyze the Diffie-Hellman key exchange process and its role in secure communications.
2. Implement secure key generation and exchange between two parties.

A.4 Theory:

- **Diffie-Hellman Key Exchange Algorithm**
- The Diffie-Hellman Key Exchange (DHKE) algorithm, proposed by Whitfield Diffie and Martin Hellman in 1976, is a method of securely exchanging cryptographic keys over an insecure communication channel. It enables two parties to jointly establish a shared secret key, which can be used for encrypting and decrypting messages.
- **Working Principle:**
  The security of the Diffie-Hellman algorithm relies on the difficulty of solving the **discrete logarithm problem**, which makes it computationally infeasible for an attacker to determine the shared secret key without the private keys of the communicating parties.

Applications of Diffie-Hellman Key Exchange:

1. **Secure Communication Protocols:**
   - Used in protocols like HTTPS, TLS/SSL, and VPNs to establish encrypted sessions.
2. **Key Agreement for Symmetric Cryptography:**
   - Enables secure distribution of symmetric keys for encryption algorithms like AES or DES.
3. **Digital Signature Schemes:**
   - Often used with RSA or ECC for secure authentication and integrity verification.
4. **Blockchain and Cryptocurrencies:**
   - Ensures secure key exchanges in decentralized networks.
5. **Internet of Things (IoT):**
   - Facilitates secure communication between connected devices.

Limitations of Diffie-Hellman Algorithm:

1. **Man-in-the-Middle Attack (MITM):**
   - Without authentication mechanisms, an attacker can impersonate one of the parties and intercept the key exchange.
   - Solution: Use digital certificates or pre-shared keys for verification.
2. **Computational Overhead:**
   - The algorithm can be computationally intensive for devices with low processing power.
3. **No Authentication:**
   - It does not authenticate the parties involved in the exchange.

▪ ALGORITHM:

1. Both parties agree on a large prime number p and a primitive root g modulo p.

2. Each party selects a private key:

   a. Alice selects a private key a.

   b. Bob selects a private key b.

3. Each party computes their public key:

   a. Alice computes A=gamod pA = g^a \mod pA=gamodp.

   b. Bob computes B=gbmod pB = g^b \mod pB=gbmodp.

4. The public keys AAA and BBB are exchanged between Alice and Bob over an insecure channel.
5. Both parties compute the shared secret key:
    a. Alice computes S=Bamod pS = B^a \mod pS=Bamodp.
    b. Bob computes S=Abmod pS = A^b \mod pS=Abmodp.

Since S=gabmod pS = g^{ab} \mod pS=gabmodp, both computations yield the same shared secret key.

EXAMPLE:

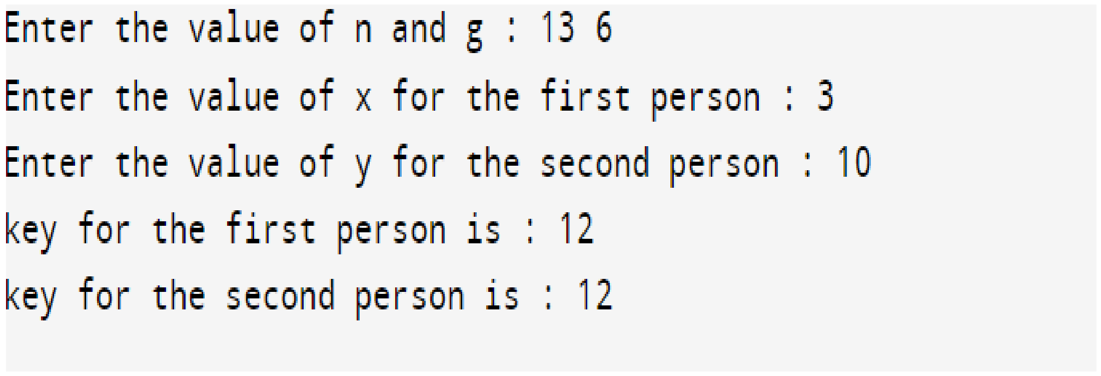
- Agree on p=23p = 23p=23 (prime number) and g=5g = 5g=5 (primitive root modulo 23).
- Alice chooses a=6a = 6a=6 (private key).
    - Computes A=56mod 23=8A = 5^6 \mod 23 = 8A=56mod23=8.
- Bob chooses b=15b = 15b=15 (private key).
    - Computes B=515mod 23=19B = 5^{15} \mod 23 = 19B=515mod23=19.
- Exchange public keys A=8A = 8A=8 and B=19B = 19B=19.
- Compute the shared secret key:
    - Alice: S=Bamod p=196mod 23=2S = B^a \mod p = 19^6 \mod 23 = 2S=Bamodp=196mod23=2.
    - Bob: S=Abmod p=815mod 23=2S = A^b \mod p = 8^{15} \mod 23 = 2S=Abmodp=815mod23=2.
- The shared secret key is S=2S = 2S=2.

Code:

```
#include<stdio.h>
```

```
long long int power(int a,int b,int mod)
{
 long long int t;
 if(b==1)
  return a;
 t=power(a,b/2,mod);
 if(b%2==0)
  return (t*t)%mod;
 else
  return (((t*t)%mod)*a)%mod;
}
long long int calculateKey(int a,int x,int n)
{
 return power(a,x,n);
}
int main()
{
 int n,g,x,a,y,b;
 printf("Enter the value of n and g : ");
 scanf("%d%d",&n,&g);
 printf("Enter the value of x for the first person : ");
 scanf("%d",&x);
 a=power(g,x,n);
 printf("Enter the value of y for the second person : ");
 scanf("%d",&y);
 b=power(g,y,n);
 printf("key for the first person is : %lld\n",power(b,x,n));
 printf("key for the second person is : %lld\n",power(a,y,n));
 return 0;
}
```

Output:



```
Enter the value of n and g : 13 6
Enter the value of x for the first person : 3
Enter the value of y for the second person : 10
key for the first person is : 12
key for the second person is : 12
```

PART B

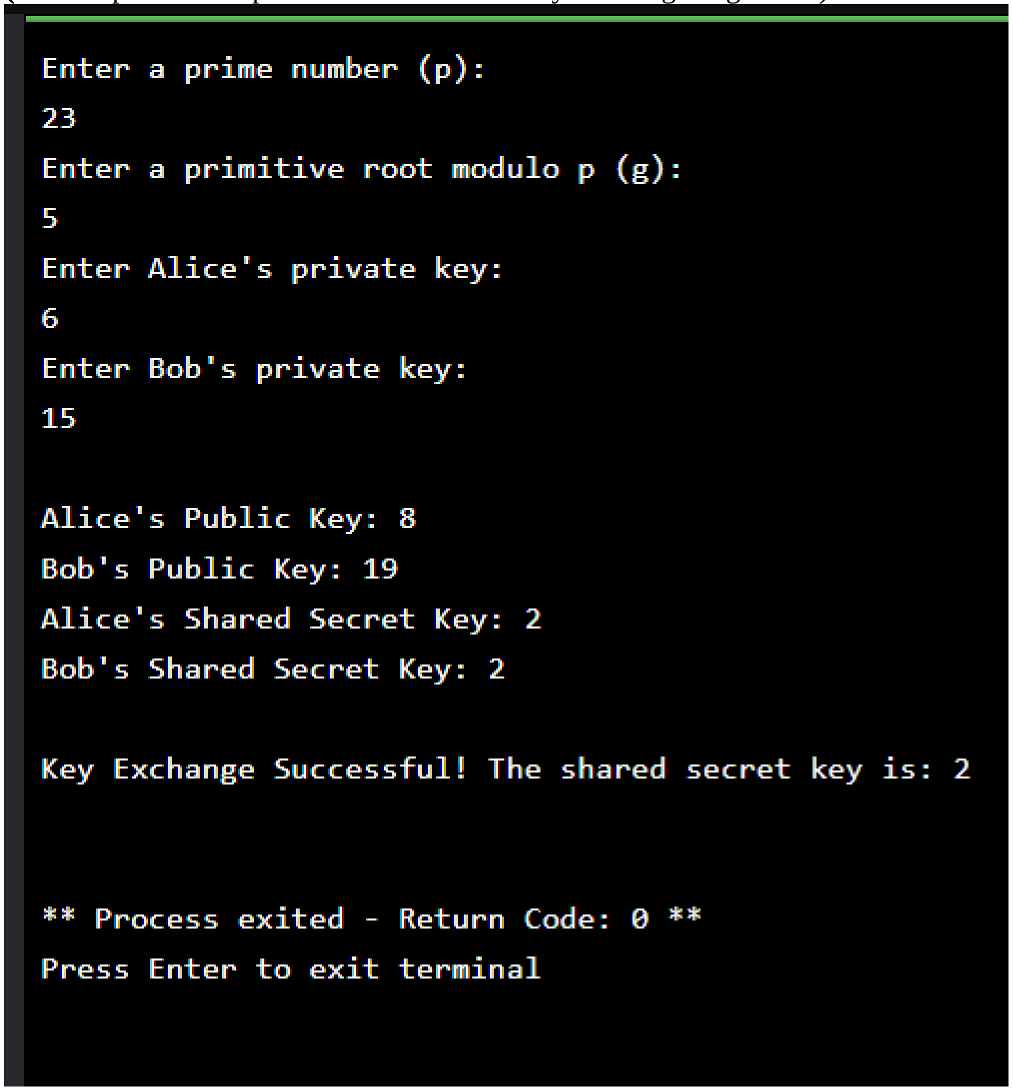
(PART B : TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)

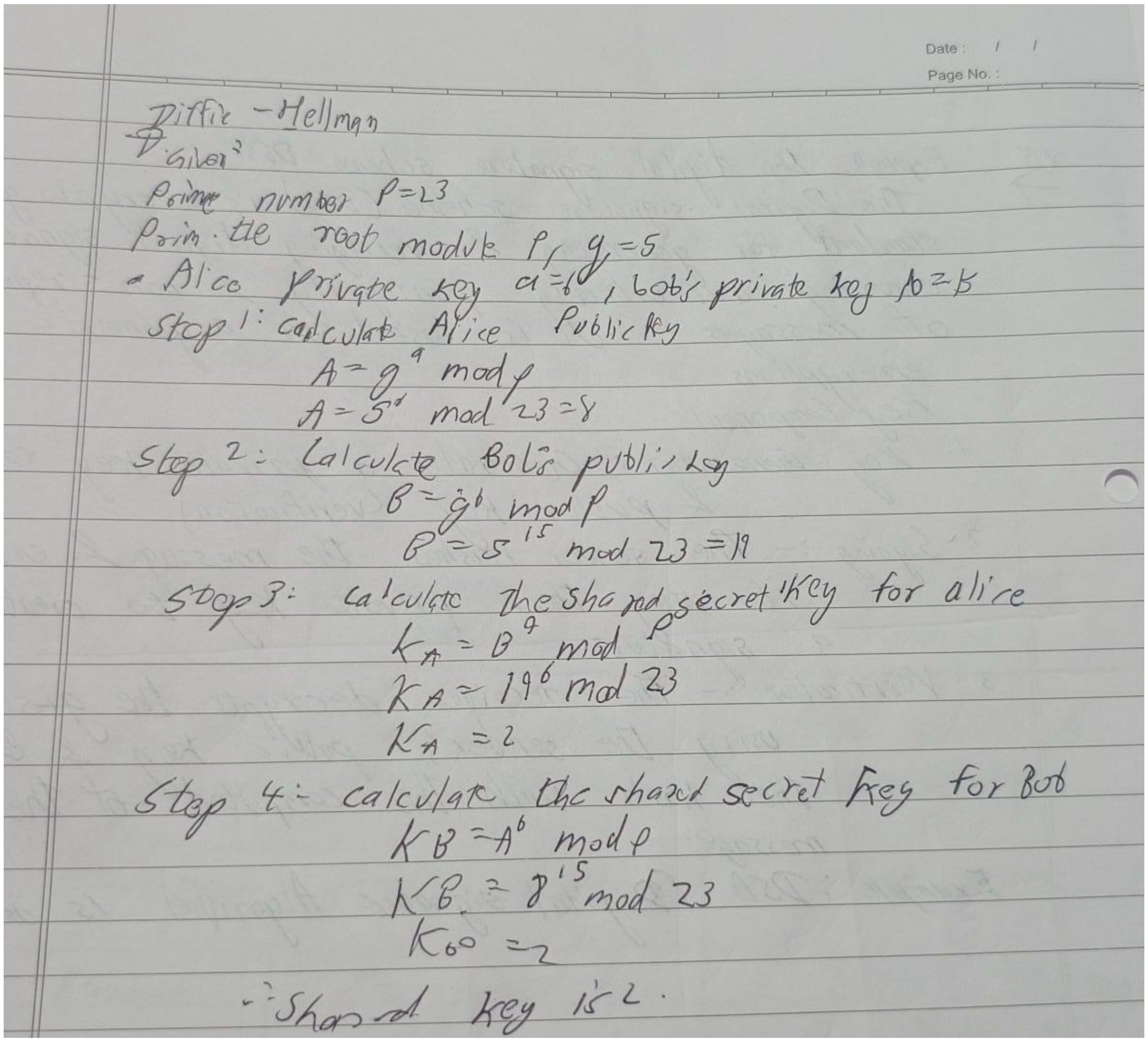
Roll No.B30	**Name: Pranjal Bhatt**
Class :COMPs TE B	**Batch :B2**
Date of Experiment:	**Date of Submission**
Grade :	

B.1 Output of Diffie-Hellman Key Exchange Algorithm:

(Add snapshot of output of Diffie-Hellman Key Exchange Algorithm)



```
Enter a prime number (p):
23
Enter a primitive root modulo p (g):
5
Enter Alice's private key:
6
Enter Bob's private key:
15

Alice's Public Key: 8
Bob's Public Key: 19
Alice's Shared Secret Key: 2
Bob's Shared Secret Key: 2

Key Exchange Successful! The shared secret key is: 2


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```


Date: / /
Page No.:

Diffie -Hellman
Given:
Prime number $p=23$
Prim. tie root module p, $g=5$
Alice private key $a=6$, bob's private key $b=15$
Step 1: calculate Alice Public key
$A = g^a \bmod p$
$A = 5^6 \bmod 23 = 8$
Step 2: Calculate Bob's public key
$B = g^b \bmod p$
$B = 5^{15} \bmod 23 = 19$
Step 3: calculate the shared secret key for alice
$K_A = B^a \bmod p$
$K_A = 19^6 \bmod 23$
$K_A = 2$
Step 4: calculate the shared secret key for Bob
$K_B = A^b \bmod p$
$K_B = 8^{15} \bmod 23$
$K_B = 2$
∴ Shared key is 2.

B.2 Source Code of Diffie-Hellman Key Exchange Algorithm:

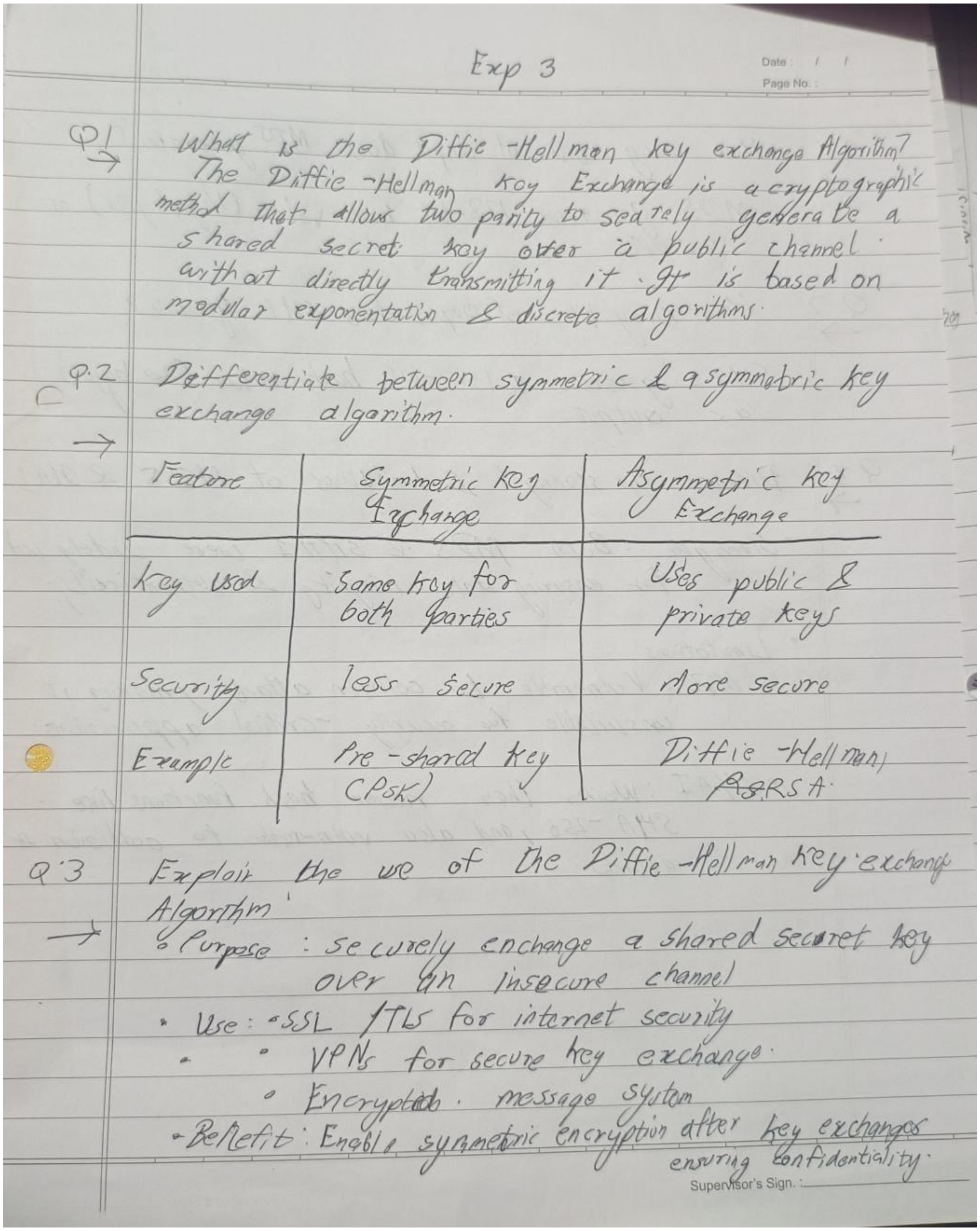
•

```
# Diffie-Hellman Key Exchange Algorithm with User Input

# Step 1: Take user input for prime number and primitive root modulo p
p = int(input("Enter a prime number (p): "))
g = int(input("Enter a primitive root modulo p (g): "))

# Step 2: Alice and Bob generate their private keys
alice_private = int(input("Enter Alice's private key: "))
bob_private = int(input("Enter Bob's private key: "))

# Step 3: Alice and Bob compute their public keys
alice_public = (g ** alice_private) % p
bob_public = (g ** bob_private) % p
```

```
# Step 4: Alice and Bob exchange their public keys

# Step 5: Alice computes the shared secret key using Bob's public key
alice_shared_key = (bob_public ** alice_private) % p

# Step 6: Bob computes the shared secret key using Alice's public key
bob_shared_key = (alice_public ** bob_private) % p

# Step 7: The shared secret key should be the same for both Alice and Bob
print("\nAlice's Public Key:", alice_public)
print("Bob's Public Key:", bob_public)
print("Alice's Shared Secret Key:", alice_shared_key)
print("Bob's Shared Secret Key:", bob_shared_key)

if alice_shared_key == bob_shared_key:
    print("\nKey Exchange Successful! The shared secret key is:", alice_shared_key)
else:
    print("\nKey Exchange Failed!")
```

B.3 Question of Curiosity: *(At least 3 questions should be handwritten)*

1. What is the Diffie-Hellman Key Exchange Algorithm?



Exp 3

Date : / /

Page No. :

Q1 What is the Diffie-Hellman key exchange Algorithm?

→ The Diffie-Hellman key Exchange is a cryptographic method that allow two parity to secrely generate a shared secret key over a public channel without directly transmitting it. It is based on modular exponentation & discrete algorithms.

Q.2 Differentiate between symmetric & asymmetric key exchange algorithm.

→

Feature	Symmetric key Exchange	Asymmetric key Exchange
key used	Same key for both parties	Uses public & private keys
Security	less secure	More secure
Example	Pre-shared key (PSK)	Diffie-Hellman, ~~RS~~ RSA.

Q.3 Explain the use of the Diffie-Hellman key exchange Algorithm.

→ ◦ Purpose : securely enchange a shared secret key over an insecure channel

- Use : ◦ SSL / TLS for internet security
  - ◦ VPNs for secure key exchange.
  - ◦ Encrypted. message system
- Benefit : Enable symmetric encryption after key exchanges ensuring confidentiality.

Supervisor's Sign. :

2. List and explain different possible attacks on Diffie-Hellman.

1. Man-in-the-Middle Attack (MITM)

An attacker intercepts and modifies the messages between Alice and Bob, making them think they're securely communicating with each other while the attacker controls the exchange.

Prevention: Use digital signatures or certificates for verification.

---

2. Pre-computed Logarithm (Logjam Attack)

Weak primes or reused values allow attackers to precompute values, making it easier to calculate the private key.

Prevention: Use large primes (2048-bit or higher).

---

3. Side-Channel Attacks

Attackers can deduce private keys by analyzing physical leaks, such as time or power usage during key exchange.

Prevention: Use constant-time algorithms and protect against side channels.

---

4. Small Subgroup Attack

Weak primes can cause private keys to belong to small groups, making it easier for attackers to break the key.

Prevention: Ensure `p` is a safe prime.

---

5. Replay Attack

Attackers can intercept and resend messages to mislead the parties into reusing keys.

Prevention: Use nonces or timestamps for uniqueness.

6. Chosen-Ciphertext Attack (CCA)

The attacker uses access to decrypted messages to infer private keys.

Prevention: Use authenticated encryption methods.

---

7. Key Reuse and Weak Key Selection

Reusing keys or choosing weak keys can make the key exchange vulnerable.

Prevention: Generate new strong keys for each exchange.

These measures help maintain the security of Diffie-Hellman key exchanges.

3. Explain the concept of Perfect Forward Secrecy (PFS) in Diffie-Hellman.

Perfect Forward Secrecy (PFS) ensures that even if a private key (or a server's private key) is compromised in the future, past communication sessions remain secure. In the context of **Diffie-Hellman**, PFS means that the session keys generated during each key exchange are not derived from long-term private keys, ensuring that each session key is unique and independent.

Key points about PFS in Diffie-Hellman:

- ◆ **Session Key Independence**: In Diffie-Hellman with PFS, the session key is derived from the ephemeral (temporary) private keys. These are generated for each session and discarded after the session ends.
- ◆ **Compromise Resistance**: Even if an attacker later gains access to the server's private key, they cannot decrypt past communications because the session keys were never stored or related to the server’s long-term private key.
- ◆ **Ephemeral Diffie-Hellman (DHE)**: A variant of Diffie-Hellman, **Ephemeral Diffie-Hellman (DHE)**, is used to achieve PFS. In DHE, both parties use temporary keys that are discarded after the session, ensuring forward secrecy.

Example:

- Alice and Bob perform a Diffie-Hellman exchange with ephemeral keys.
- Even if the attacker later steals Alice’s or Bob’s private key, they cannot decrypt past

messages because the session keys were discarded after use.

In summary, **PFS** makes Diffie-Hellman more secure by ensuring that past communication is protected, even if long-term keys are compromised in the future.

4. How does the Diffie-Hellman algorithm ensure secure key exchange without sharing the private key?

The **Diffie-Hellman algorithm** ensures secure key exchange without directly sharing private keys through the use of mathematical operations involving **public keys**, **modular arithmetic**, and **exponentiation**. Here's how it works:

1. Private Key Generation:

- Both parties (Alice and Bob) each generate their own **private key**, which is kept secret.
  - Let Alice's private key be `a`.
  - Let Bob's private key be `b`.

2. Public Key Calculation:

- Each party computes their **public key** using the shared base `g` and prime modulus `p` (which are public):
  - Alice's public key: A=gamod pA = g^a \mod p
  - Bob's public key: B=gbmod pB = g^b \mod p
- These public keys are exchanged between Alice and Bob.

3. Shared Secret Calculation:

- After receiving each other's public keys, Alice and Bob compute the shared secret key independently:
  - Alice computes the shared secret key: SA=Bamod pS_A = B^a \mod p
  - Bob computes the shared secret key: SB=Abmod pS_B = A^b \mod p

  Due to the properties of modular exponentiation, both Alice and Bob end up with the same shared secret key:

  - SA=gabmod pS_A = g^{ab} \mod p
  - SB=gabmod pS_B = g^{ab} \mod p

This key exchange happens without either party revealing their private key. The security lies in the **difficulty of computing discrete logarithms**, which means that even though the public keys `A` and `B` are exchanged, it is computationally infeasible for an attacker to reverse-engineer the private keys `a` or `b` from these public values.

Key Points:

- **Private keys** are never transmitted over the network.
- Only **public keys** (`A` and `B`) are exchanged.
- The **shared secret key** is calculated independently by both parties using the other party's public key.
- The security of the exchange is based on the **difficulty of calculating discrete logarithms**.

Thus, Diffie-Hellman enables secure key exchange because the shared secret is established without the need to share private keys, and even if someone intercepts the public keys, they cannot easily derive the shared secret without solving a computationally hard problem.

B.4 Conclusion:

The **Diffie-Hellman Key Exchange algorithm** allows two parties to securely establish a shared secret key over an insecure channel without exchanging private keys. By using modular arithmetic and exponentiation, it ensures that even if public keys are intercepted, the shared secret remains secure. This experiment highlighted the importance of mathematical principles in cryptography and demonstrated how Diffie-Hellman facilitates secure communication by preventing the direct exchange of private keys.