|  |  |
| --- | --- |
| **Academic Year: 2024-25** | **Programme: BTECH-Cyber (CSE)** |
| **Year: 2nd** | **Semester: IV** |
| **Student Name: Jal Bafana** | **Batch: K1** |
| **Roll No: K005** | **Date of experiment: 30.01.2025** |
| **Faculty: Rejo Mathew** | **Signature with Date: 30.01.2025** |

**Experiment 3: Diffie-Hellman Key Exchange**

**Aim:** Write a program to implement Diffie-Hellman Key exchange.

**Learning Outcomes:**

After completion of this experiment, student should be able to

1. Differentiate between symmetric and asymmetric key cryptography.
2. Describe working of Diffie-Hellman key exchange.
3. Understand application of Diffie-Hellman along with its advantage and limitations.

**Theory:**

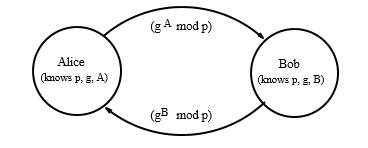
It is a method of securely exchanging cryptographic keys over a public channel and was one of the first public-key protocols. It is one of the earliest practical examples of public key exchange implemented within the field of cryptography.

Traditionally, secure encrypted communication between two parties required that they first exchange keys by some secure physical channel, such as paper key lists transported by a trusted courier. This method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

It is used to secure a variety of Internet services. Although Diffie–Hellman key agreement itself is a non-authenticated key-agreement protocol, it provides the basis for a variety of authenticated protocols, and is used to provide forward secrecy in Transport Layer Security's ephemeral modes (referred to as EDH or DHE depending on the cipher suite).

The method was followed shortly afterwards by RSA, an implementation of public-key cryptography using asymmetric algorithms

**Algorithm:**



1. Alice and Bob agree on a prime number p and a base g
2. Alice chooses a secret number a, and sends Bob (ga mod p)
3. Bob chooses a secret number b, and sends Bob (gb mod p)
4. Alice computes ((g b modp) a mod p)
5. Bob computes ((g a modp) b mod p)
6. Alice and Bob can use this number as key

**Code: *type or copy your completed working code here***

#include <iostream>

using namespace std;

bool isPrimitiveRoot(int g, int p)

{

int visited[p];

for (int i = 0; i < p; i++)

{

visited[i] = -1;

}

for (int n = 1; n < p; n++)

{

int power = 1;

for (int i = 0; i < n; i++)

{

power = (power \* g) % p;

}

if (visited[power] != -1)

{

return false;

}

visited[power] = n;

}

return true;

}

int main()

{

int p,g,i;

cout<<"Enter a prime number (p): ";

cin>>p;

cout<<"Enter a base number (g): ";

cin>>g;

if(!isPrimitiveRoot(g, p))

{

cout<<"g is NOT a primitive root of p"<<endl;

return 0;

}

cout<<"g is a primitive root of p" << endl;

int a,b;

cout<<"Enter Alice's secret number (a): ";

cin>>a;

cout<<"Enter Bob's secret number (b): ";

cin>>b;

int A=1;

for(i=0; i<a; i++)

{

A = (A \* g) % p;

}

cout<<"Alice sends A to Bob: "<<A<<endl;

int B = 1;

for(i=0; i<b; i++)

{

B = (B \* g) % p;

}

cout<<"Bob sends B to Alice: "<<B<<endl;

int K\_Alice = 1;

for (int i = 0; i < a; i++)

{

K\_Alice = (K\_Alice \* B) % p;

}

cout<<"Alice computes shared key: "<<K\_Alice<<endl;

int K\_Bob = 1;

for(i=0; i<b; i++)

{

K\_Bob = (K\_Bob \* A) % p;

}

cout<<"Bob computes shared key: "<<K\_Bob<<endl;

if (K\_Alice == K\_Bob)

{

cout<<"The shared secret key is: "<<K\_Alice<<endl;

}

else

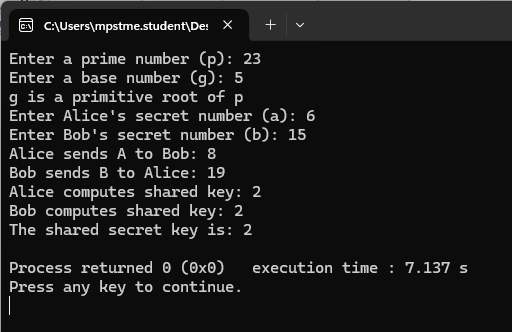
{

cout<<"Error: Keys do not match"<<endl;

}

return 0;

}



**Questions:**

1. Write a short note about Discrete Log problem

The **Discrete Logarithm Problem (DLP)** is a mathematical problem that forms the foundation for many cryptographic algorithms, including Diffie-Hellman and Elliptic Curve Diffie-Hellman (ECDH). It involves finding the exponent xxx in the equation gxmod  p=yg^x \mod p = ygxmodp=y, where:

* g is a primitive root modulo ppp,
* ppp is a large prime number,
* yyy is the result of the exponentiation gxmod  pg^x \mod pgxmodp.

The challenge of solving this problem is that while it is easy to compute gxmod  pg^x \mod pgxmodp for a known xxx, it is computationally hard to determine xxx given ggg, ppp, and yyy. This one-way function, based on the difficulty of solving the discrete logarithm, underpins the security of many cryptographic systems.

1. **What modifications does Elliptic Curve Diffie-Hellman (ECDH) introduce to the original Diffie-Hellman protocol?**

* Elliptic Curve Diffie-Hellman (ECDH) introduces several modifications to the original Diffie-Hellman (DH) protocol:
* Elliptic Curve Cryptography (ECC): Instead of using the multiplicative group of integers modulo a prime (as in DH), ECDH uses the group of points on an elliptic curve over a finite field. This provides the same level of security with much smaller key sizes, making it more efficient.
* Smaller Key Sizes: ECDH can achieve the same cryptographic security as traditional DH but with much shorter keys, leading to faster computations and reduced storage requirements.
* Higher Security: The mathematical structure of elliptic curves makes certain attacks (like the number field sieve) much more difficult, offering better security with smaller computational overhead.
* Efficiency: Due to the smaller key sizes and faster computations, ECDH is more efficient, especially for devices with limited resources, such as mobile phones and IoT devices.

1. Key Exchange algorithms are vulnerable to which type of attacks. How to prevent?

Key exchange algorithms, like Diffie-Hellman and ECDH, are vulnerable to several types of attacks:

* Man-in-the-Middle Attack (MITM): An attacker can intercept and alter messages exchanged between the two parties. This is particularly dangerous in Diffie-Hellman, where the exchange is not authenticated.
* Prevention: Use digital signatures or certificates to authenticate the exchanged public keys. Public key infrastructure (PKI) and trusted certificate authorities (CAs) can help prevent MITM attacks.
* Replay Attack: An attacker might intercept a valid key exchange message and resend it to the recipient, trying to establish a connection.
* Prevention: Use unique, non-repeating session identifiers or timestamps, and incorporate them into the key exchange process to ensure messages are fresh and not reused.
* Weak Parameters (Small Key Sizes): Using weak or small prime numbers in Diffie-Hellman can make the key exchange vulnerable to attacks like brute-force or the number field sieve.
* Prevention: Use sufficiently large prime numbers for the group and ensure that the key exchange protocol uses parameters that are resistant to known attacks.
* Side-Channel Attacks: These attacks exploit physical characteristics (like timing or power consumption) of a cryptographic device to learn information about the secret key.
* Prevention: Implement constant-time algorithms and side-channel resistance techniques, such as masking or noise injection.

1. Explain the difference between **static Diffie-Hellman (DH)** and **ephemeral Diffie-Hellman (DHE)**.

* **Static Diffie-Hellman (DH):** In static DH, each participant uses a fixed, long-term private key to generate a public key. The same public key is reused for multiple sessions, making it more vulnerable to certain attacks (e.g., long-term key exposure risks).
* **Ephemeral Diffie-Hellman (DHE):** In ephemeral DH, each participant generates a new, temporary private key for every session. This ensures that even if the session key is compromised, previous session keys remain secure, as the key material is discarded after the session. DHE provides forward secrecy, meaning that even if long-term keys are compromised, past communications remain secure.

**Conclusion:** *This lab covered Diffie-Hellman (DH) and Elliptic Curve Diffie-Hellman (ECDH) for secure key exchange, highlighting efficiency and security measures like authentication and ephemeral keys.*