Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

Diffie-Hellman key exchange

Theory:

The Diffie-Hellman key exchange, often simply referred to as Diffie-Hellman, is a public-key cryptography algorithm that allows two parties to securely exchange cryptographic keys over an insecure communication channel. The fundamental idea behind Diffie-Hellman is to enable two parties to agree on a shared secret key without having to transmit the key itself. Here's the theory behind the Diffie-Hellman key exchange:

1. Key Elements:

   - Public Parameters: There are two publicly known values in the system: a large prime number, often denoted as "p," and a base value, typically denoted as "g." These parameters are shared between the parties and do not need to be kept secret.

   - Private Keys: Each party (Alice and Bob) generates their own private key, which is kept secret and never shared with anyone. Alice's private key is typically denoted as "a," and Bob's private key as "b."

2. Key Generation:

   - Alice and Bob independently select their private keys, "a" and "b," respectively. These private keys are random and secret.

3. Public Key Calculation:

   - Alice calculates her public key "A" using the following formula:
    ```

    A = g^a mod p
    ```
   - Bob calculates his public key "B" in a similar manner:

```
B = g^b mod p
```

4. Exchange Public Keys:

- Alice sends her public key "A" to Bob.
- Bob sends his public key "B" to Alice.

5. Shared Secret Calculation:

- Both Alice and Bob use the other party's public key and their own private key to calculate the same shared secret key.

- Alice calculates the shared secret "s" as follows:

```
s = B^a mod p
```

- Bob calculates the shared secret "s" in a similar manner:

```
s = A^b mod p
```

- Due to the mathematical properties of modular exponentiation, both Alice and Bob will arrive at the same shared secret key "s." This shared secret can be used for encryption and decryption.

6. Security:

Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

- The security of the Diffie-Hellman key exchange relies on the difficulty of the discrete logarithm problem. In other words, it is computationally difficult to determine the private keys "a" and "b" from the exchanged public keys "A" and "B" when the values of "p" and "g" are sufficiently large.

7. Perfect Forward Secrecy:

- One of the notable advantages of Diffie-Hellman is that it provides perfect forward secrecy. Even if an attacker were to compromise the private keys at some point in the future, they would not be able to retroactively decrypt past communications since those communications were encrypted with different shared secrets.

The Diffie-Hellman key exchange is widely used in secure communication protocols and is an essential component of modern cryptography, including secure internet connections (e.g., TLS/SSL) and secure messaging systems. However, it is important to choose appropriate values for "p" and "g" and to use larger key sizes for better security in practice, as the security of Diffie-Hellman depends on the difficulty of the discrete logarithm problem.

Code:

Alice:

```
#include <iostream>
#include <cmath>
#include <winsock2.h>

long long p = 17; // Large prime number (public)
long long alpha = 5;  // Primitive root modulo p (public)

long long powM(long long a, long long b, long long n){
    if (b == 1){
        return a % n;
    }
    long long x = powM(a, b / 2, n);
    x = (x * x) % n;
    if (b % 2){
        x = (x * a) % n;
    }
    return x;
}

int main() {
    WSADATA wsaData;
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {
```

Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

```
        std::cerr << "Failed to initialize Winsock" << std::endl;
        return -1;
    }

    SOCKET clientSocket;
    struct sockaddr_in serverAddress;

    clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (clientSocket == INVALID_SOCKET) {
        std::cerr << "Error creating socket" << std::endl;
        WSACleanup();
        return -1;
    }

    serverAddress.sin_family = AF_INET;
    serverAddress.sin_port = htons(8080);
    serverAddress.sin_addr.s_addr = inet_addr("127.0.0.1"); //Localhost

    if (connect(clientSocket, (struct sockaddr*)&serverAddress,
sizeof(serverAddress)) == SOCKET_ERROR) {
        std::cerr << "Error connecting to Bob" << std::endl;
        closesocket(clientSocket);
        WSACleanup();
        return -1;
    }

    int xa = 4; // Alice's private key

    // Alice computes A = (alpha^xa) % p
    int A = powM(alpha, xa, p);
    std::cout << "Alice computes A: " << A << std::endl;

    // Send Alice's public value A to Bob
    send(clientSocket, (char*)&A, sizeof(A), 0);
    std::cout << "Sent Alice's public value A to Bob" << std::endl;

    // Receive Bob's public value B
    int B;
    recv(clientSocket, (char*)&B, sizeof(B), 0);
    std::cout << "Received Bob's public value B: " << B << std::endl;

    // Calculate the shared secret key
    int shared_key_alice = powM(B, xa, p);
    std::cout << "Shared key calculated by Alice: " << shared_key_alice <<
std::endl;

    closesocket(clientSocket);
    WSACleanup();
```

Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

```
    return 0;
}
```

Bob:

```
#include <iostream>
#include <cmath>
#include <winsock2.h>

long long p = 17; // Large prime number (public)
long long alpha = 5;  // Primitive root modulo p (public)

long long powM(long long a, long long b, long long n){
    if (b == 1){
        return a % n;
    }
    long long x = powM(a, b / 2, n);
    x = (x * x) % n;
    if (b % 2){
        x = (x * a) % n;
    }
    return x;
}

int main() {
    WSADATA wsaData;
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {
        std::cerr << "Failed to initialize Winsock" << std::endl;
        return -1;
    }

    SOCKET serverSocket;
    struct sockaddr_in serverAddress;
    SOCKET clientSocket;
    struct sockaddr_in clientAddress;

    serverSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (serverSocket == INVALID_SOCKET) {
        std::cerr << "Error creating socket" << std::endl;
        WSACleanup();
        return -1;
    }

    serverAddress.sin_family = AF_INET;
    serverAddress.sin_port = htons(8080);
    serverAddress.sin_addr.s_addr = INADDR_ANY;
```

Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

```
    if (bind(serverSocket, (struct sockaddr*)&serverAddress,
sizeof(serverAddress)) == SOCKET_ERROR) {
        std::cerr << "Error binding to port" << std::endl;
        closesocket(serverSocket);
        WSACleanup();
        return -1;
    }

    listen(serverSocket, 1);
    std::cout << "Bob is waiting for Alice to connect..." << std::endl;

    int clientAddress_size = sizeof(clientAddress);
    clientSocket = accept(serverSocket, (struct sockaddr*)&clientAddress,
&clientAddress_size);

    if (clientSocket == INVALID_SOCKET) {
        std::cerr << "Error accepting the connection" << std::endl;
        closesocket(serverSocket);
        WSACleanup();
        return -1;
    }

    int xb = 6; // Bob's private key

    // Receive Alice's public value A
    int A;
    recv(clientSocket, (char*)&A, sizeof(A), 0);
    std::cout << "Received Alice's public value A: " << A << std::endl;

    // Bob computes B = (alpha^xb) % p
    int B = powM(alpha, xb, p);
    std::cout << "Bob computes B: " << B << std::endl;

    // Send Bob's public value B to Alice
    send(clientSocket, (char*)&B, sizeof(B), 0);
    std::cout << "Sent Bob's public value B to Alice" << std::endl;

    // Calculate the shared secret key
    int shared_key_bob = powM(A, xb, p);
    std::cout << "Shared key calculated by Bob: " << shared_key_bob <<
std::endl;

    closesocket(serverSocket);
    closesocket(clientSocket);
    WSACleanup();

    return 0;
```

Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

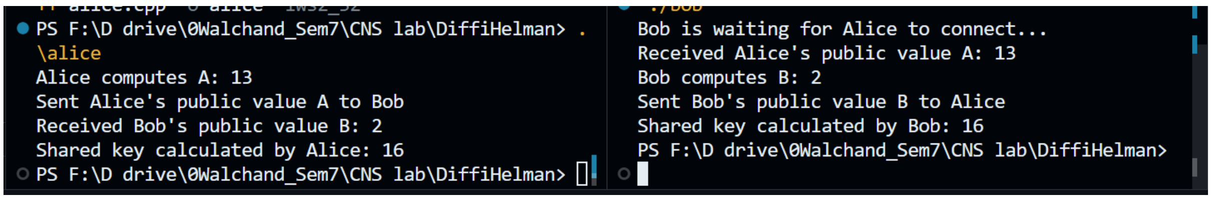
```
}
```

Screenshot:

g++ alice.cpp -o alice -lws2_32

g++ bob.cpp -o bob -lws2_32

.\alice   .\bob



```
PS F:\D drive\0Walchand_Sem7\CNS lab\DiffiHelman> .
\alice
Alice computes A: 13
Sent Alice's public value A to Bob
Received Bob's public value B: 2
Shared key calculated by Alice: 16
PS F:\D drive\0Walchand_Sem7\CNS lab\DiffiHelman>
```

```
Bob is waiting for Alice to connect...
Received Alice's public value A: 13
Bob computes B: 2
Sent Bob's public value B to Alice
Shared key calculated by Bob: 16
PS F:\D drive\0Walchand_Sem7\CNS lab\DiffiHelman>
```