Assignment 7

Diffie Hellman

Name: Shreeshail Mahajan.

PRN: 2020BTECS00057

Aim: The aim of the Diffie-Hellman key exchange protocol is to enable secure communication over an untrusted or potentially eavesdropped channel by allowing two parties to establish a shared secret key without directly transmitting that key.

Theory:

The Diffie-Hellman key exchange is a foundational cryptographic protocol that enables two parties to securely establish a shared secret key over an untrusted communication channel. This shared secret key can subsequently be used for encryption and decryption to ensure data confidentiality. The theory behind the Diffie-Hellman protocol can be explained as follows:

1. Modular Arithmetic:

The foundation of Diffie-Hellman lies in modular arithmetic. Given a large prime number \(p\) and a primitive root modulo \(p\), denoted as \(g\), both of which are publicly known, parties involved in the key exchange perform calculations in modulo \(p\).

2. Public Parameters:

The prime number \(p\) and primitive root \(g\) are considered public parameters. These parameters can be shared openly, and their security doesn't rely on their secrecy but on the difficulty of solving the discrete logarithm problem.

3. Key Generation:

Each party, say Alice and Bob, chooses their own private key, which is a randomly selected integer. Alice selects \(a\), and Bob selects \(b\). These private keys must be kept secret.

4. Public Values:

Alice and Bob then compute their public values as follows:

- Alice computes \(A = g^a \mod p\) and shares \(A\) publicly.

- Bob computes \(B = g^b \mod p\) and shares \(B\) publicly.

Both \(A\) and \(B\) are exchanged openly, and anyone can observe them.

5. Shared Secret Key:

To derive the shared secret key, Alice uses her private key and Bob's public value:

- Alice computes the shared secret key as \(S = B^a \mod p\).

Bob does the same using his private key and Alice's public value:

- Bob computes the shared secret key as \(S = A^b \mod p\).

Importantly, due to the properties of modular arithmetic, \(S\) calculated by Alice and \(S\) calculated by Bob are the same. This shared secret key can be used for encryption and decryption.

6. Security:

The security of Diffie-Hellman relies on the difficulty of solving the discrete logarithm problem. Given \(p\), \(g\), and \(A\) (or \(B\)), it is computationally infeasible to determine the private keys \(a\) and \(b\) from which \(A\) (or \(B\)) was derived. This makes it secure against eavesdroppers who can observe \(A\) and \(B\) but cannot easily compute the shared secret key.

7. Refreshment and Reuse:

For security reasons, it is recommended to periodically refresh the private keys \(a\) and \(b) and use new values. Additionally, the same public parameters \(p\) and \(g\) can be reused for multiple key exchanges.

In summary, the Diffie-Hellman key exchange protocol allows two parties to agree upon a shared secret key for secure communication without ever transmitting the key itself. This is accomplished through modular arithmetic operations, leveraging the difficulty of the discrete logarithm problem to maintain security. The protocol is widely used in secure communication and forms the basis of many secure communication protocols and cryptographic systems.

Code:

#include <bits/stdc++.h>

using namespace std;

void file()

{

#ifndef ONLINE\_JUDGE

freopen("input.txt", "r", stdin);

freopen("output.txt", "w", stdout);

#endif

}

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;

}

bool checkPrimitiveRoot(long long alpha, long long q)

{

map<long long, int> m;

for (long long i = 1; i < q; i++)

{

long long x = powM(alpha, i, q);

//cout << x << endl;

if (m.find(x) != m.end())

return 0;

m[x] = 1;

}

return 1;

}

int main()

{

long long q, alpha;

q = 71; // A prime number q is taken

alpha = 7; // A primitive root of q

if (checkPrimitiveRoot(alpha, q) == 0)

{

cout << "alpha is not primitive root of q";

return 0;

}

else

{

cout << alpha << " is private root of " << q << endl;

}

long long xa, ya;

xa = 4; // xa is the chosen private key

ya = powM(alpha, xa, q); // public key of alice

cout << "private key of alice is " << xa << endl;

cout << "public key of alice is " << ya << endl << endl;

long long xb, yb;

xb = 3; // xb is the chosen private key

yb = powM(alpha, xb, q); // public key of bob

cout << "private key of bob is " << xb << endl;

cout << "public key of bob is " << yb << endl << endl;

//key generation

long long k1, k2;

k1 = powM(yb, xa, q); // Secret key for Alice

k2 = powM(ya, xb, q); // Secret key for Bob

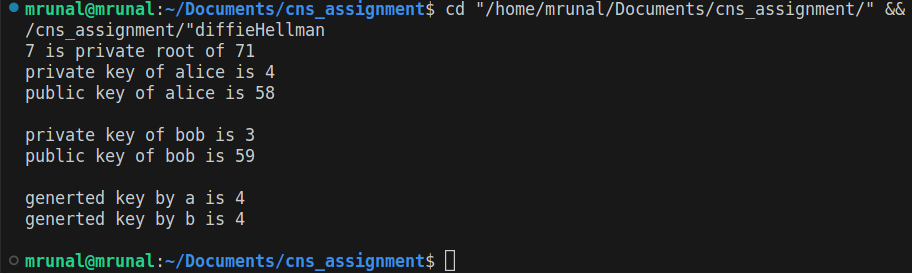
cout << "generated key by a is " << k1 << endl;

cout << "generated key by b is " << k2 << endl << endl;

return 0;

}

Output:



Conclusion:

Diffie-Hellman key exchange protocol allows two parties to agree upon a shared secret key for secure communication without ever transmitting the key itself. This is accomplished through modular arithmetic operations, leveraging the difficulty of the discrete logarithm problem to maintain security. The protocol is widely used in secure communication and forms the basis of many secure communication protocols and cryptographic systems.