**Name: Kshitija Shamrao Jadhav**

**PRN: 2019BTECS00053**

**BATCH: B6**

**Diffie Hellman**

**Diffie-Hellman algorithm**

The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

* For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables, one prime P and G (a primitive root of P) and two private values a and b.
* P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly. The opposite person receives the key and that generates a secret key, after which they have the same secret key to encrypt.

**Step by Step Explanation**

| Alice | Bob |
| --- | --- |
| Public Keys available = P, G | Public Keys available = P, G |
| Private Key Selected = a | Private Key Selected = b |
| Key generated = | Key generated = |
| Exchange of generated keys takes place | |
| Key received = y | key received = x |
| Generated Secret Key = | Generated Secret Key = |
| Algebraically, it can be shown that | |
| Users now have a symmetric secret key to encrypt | |

**Example:**

Step 1: Alice and Bob get public numbers P = 23, G = 9

Step 2: Alice selected a private key a = 4 and

Bob selected a private key b = 3

Step 3: Alice and Bob compute public values

Alice: x =(9^4 mod 23) = (6561 mod 23) = 6

Bob: y = (9^3 mod 23) = (729 mod 23) = 16

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key y =16 and

Bob receives public key x = 6

Step 6: Alice and Bob compute symmetric keys

Alice: ka = y^a mod p = 65536 mod 23 = 9

Bob: kb = x^b mod p = 216 mod 23 = 9

Step 7: 9 is the shared secret.

/\* This program calculates the Key for two persons

using the Diffie-Hellman Key exchange algorithm using C++ \*/

#include <cmath>

#include <iostream>

using namespace std;

// Power function to return value of a ^ b mod P

long long int power(long long int a, long long int b,

                    long long int P)

{

    if (b == 1)

        return a;

    else

        return (((long long int)pow(a, b)) % P);

}

// Driver program

int main()

{

    long long int P, G, x, a, y, b, ka, kb;

    // Both the persons will be agreed upon the

    // public keys G and P

    P = 23; // A prime number P is taken

    cout << "The value of P : " << P << endl;

    G = 9; // A primitive root for P, G is taken

    cout << "The value of G : " << G << endl;

    // Alice will choose the private key a

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

    cout << "The private key a for Alice : " << a << endl;

    x = power(G, a, P); // gets the generated key

    // Bob will choose the private key b

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

    cout << "The private key b for Bob : " << b << endl;

    y = power(G, b, P); // gets the generated key

    // Generating the secret key after the exchange

    // of keys

    ka = power(y, a, P); // Secret key for Alice

    kb = power(x, b, P); // Secret key for Bob

    cout << "Secret key for the Alice is : " << ka << endl;

    cout << "Secret key for the Alice is : " << kb << endl;

    return 0;

}

**OUTPUT:**

