Implementation of Diffie-Hellman Algorithm

Difficulty Level : Medium • Last Updated : 24 Oct, 2021

Background

Elliptic Curve Cryptography (ECC) is an approach to public-key cryptography, based on the algebraic structure of elliptic curves over finite fields. ECC requires a smaller key as compared to non-ECC cryptography to provide equivalent security (a 256-bit ECC security has equivalent security attained by 3072-bit RSA cryptography).

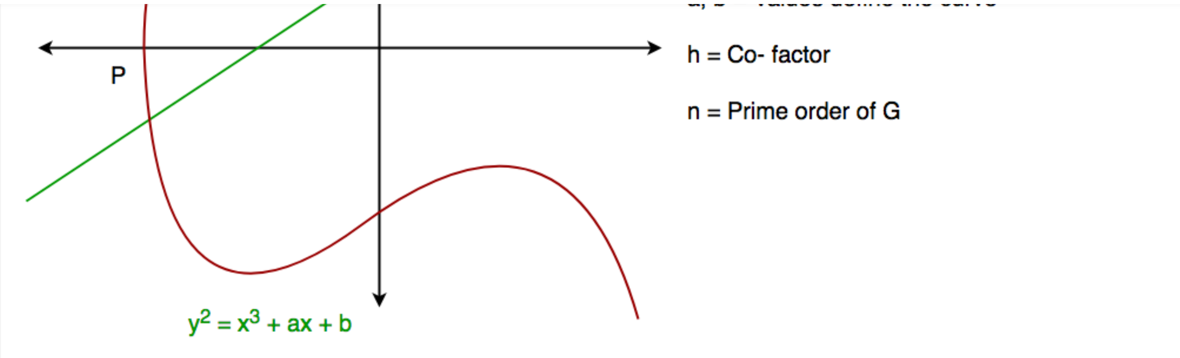
For a better understanding of Elliptic Curve Cryptography, it is very important to understand the basics of the Elliptic Curve. An elliptic curve is a planar algebraic curve defined by an equation of the form

$y^2 = x^3 + ax + b$

Where 'a' is the co-efficient of x and 'b' is the constant of the equation

The curve is non-singular; that is, its graph has no cusps or self-intersections (when the characteristic of the Co-efficient field is equal to 2 or 3).

In general, an elliptic curve looks like as shown below. Elliptic curves can intersect almost 3 points when a straight line is drawn intersecting the curve. As we can see, the elliptic curve is symmetric about the x-axis. This property plays a key role in the algorithm.



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

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

$x = G^a mod P$ $y = G^b mod P$

Exchange of generated keys takes place

Key received = y key received = x

Generated Secret Key = Generated Secret Key =

$k_a = y^a mod P$ $k_b = x^b mod P$

Algebraically, it can be shown that

$k_a = k_b$

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 =
```

```
Step 7: 9 is the shared secret.
```

Implementation:

C

```c
/* This program calculates the Key for two persons
using the Diffie-Hellman Key exchange algorithm */
#include<stdio.h>
#include<math.h>

// 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
    printf("The value of P : %lld\n", P);

    G = 9; // A primitive root for P, G is taken
    printf("The value of G : %lld\n\n", G);

    // Alice will choose the private key a
    a = 4; // a is the chosen private key
    printf("The private key a for Alice : %lld\n", a);
    x = power(G, a, P); // gets the generated key

    // Bob will choose the private key
```

```
    ka = power(y, a, P); // Secret key for Alice
    kb = power(x, b, P); // Secret key for Bob

    printf("Secret key for the Alice is : %lld\n", ka);
    printf("Secret Key for the Bob is : %lld\n", kb);

    return 0;
}
```

Java

```
// This program calculates the Key for two persons
// using the Diffie-Hellman Key exchange algorithm
class GFG{

// Power function to return value of a ^ b mod P
private static long power(long a, long b, long p)
{
    if (b == 1)
        return a;
    else
        return (((long)Math.pow(a, b)) % p);
}

// Driver code
public static void main(String[] args)
{
    long P, G, x, a, y, b, ka, kb;

    // Both the persons will be agreed upon the
    // public keys G and P

    // A prime number P is taken
    P = 23;
    System.out.println("The value of P:" + P);

    // A primitive root for P, G is taken
    G = 9;
    System.out.println("The value of G:" + G);

    // Alice will choose the private ke
```

```
        // Bob will choose the private key b
        // b is the chosen private key
        b = 3;
        System.out.println("The private key b for Bob:" + b);

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

        // 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

        System.out.println("Secret key for the Alice is:" + ka);
        System.out.println("Secret key for the Bob is:" + kb);
    }
    }

    // This code is contributed by raghav14
```

Python3

```
from random import randint

if __name__ == '__main__':

    # Both the persons will be agreed upon the
    # public keys G and P
    # A prime number P is taken
    P = 23

    # A primitive root for P, G is taken
    G = 9


    print('The Value of P is :%d'%(P))
    print('The Value of G is :%d'%(G))

    # Alice will choose the private key a
    a = 4
```

```
    b = 3
    print('The Private Key b for Bob is :%d'%(b))

    # gets the generated key
    y = int(pow(G,b,P))


    # Secret key for Alice
    ka = int(pow(y,a,P))

    # Secret key for Bob
    kb = int(pow(x,b,P))

    print('Secret key for the Alice is : %d'%(ka))
    print('Secret Key for the Bob is : %d'%(kb))
```

Javascript

```
<script>

// This program calculates the Key for two persons
// using the Diffie-Hellman Key exchange algorithm

// Power function to return value of a ^ b mod P
function power(a, b, p)
{
    if (b == 1)
        return a;
    else
        return((Math.pow(a, b)) % p);
}

// Driver code
var P, G, x, a, y, b, ka, kb;

// Both the persons will be agreed upon the
// public keys G and P

// A prime number P is taken
P = 23;
document.write("The value of P:" + P + "<br>");
```

```
// a is the chosen private key
a = 4;
document.write("The private key a for Alice:" +
               a + "<br>");

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

// Bob will choose the private key b
// b is the chosen private key
b = 3;
document.write("The private key b for Bob:" +
               b + "<br>");

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

// 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

document.write("Secret key for the Alice is:" +
               ka + "<br>");
document.write("Secret key for the Bob is:" +
               kb + "<br>");

// This code is contributed by Ankita saini

</script>
```

Output:

```
The value of P : 23
The value of G : 9

The private key a for Alice : 4
The private key b for Bob : 3
```

contribute, you can also write an article using contribute.GeeksforGeeks.org or mail your article to contribute@GeeksforGeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Previous Next

RECOMMENDED ARTICLES

Page : 1 2 3

01 Implementation of a Back-off Algorithm for CSMA/CD
17, Apr 20

05 Approaches to Information Security Implementation
26, Feb 20

03 Hellman Algorithm between Client and Server
18, Oct 18

07 UDP Server-Client implementation in C
22, Mar 18

04 Hamming code Implementation in Java
02, Aug 19

08 TCP Server-Client implementation in C
12, Sep 18

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

Easy Normal Medium Hard Expert

Improved By : nalasivam, ShJos, raghav14, ankita_saini, sooda367

Article Tags : Computer Networks

Practice Tags : Computer Networks

Improve Article Report Issue

GeeksforGeeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org