GeeksforGeeks
A computer science portal for geeks

Custom Search

Courses

Login

Write an Article

Implementation of Diffie-Hellman Algorithm

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 have an equivalent security attained by 3072-bit RSA cryptography).

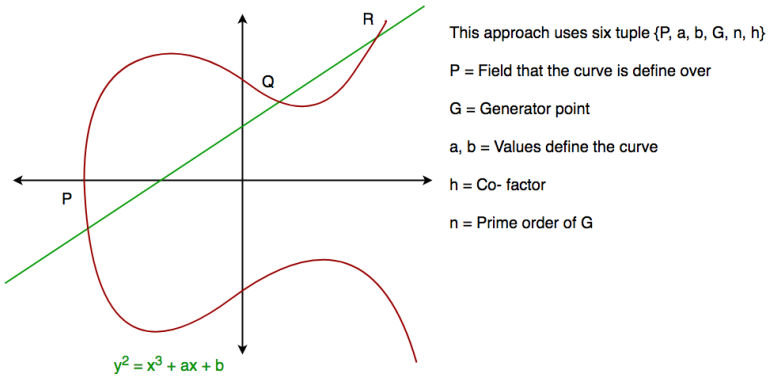
For a better understanding of Elliptic Curve Cryptography, it is very important to understand the basics of Elliptic Curve. An elliptic curve is a planar algebraic curve defined by an equation of the form

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

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

Most Accurate
Email Verifier

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 could intersect atmost 3 points when a straight line is drawn intersecting the curve. As we can see that elliptic curve is symmetric about the x-axis, this property plays a key role in the algorithm.



This approach uses six tuple {P, a, b, G, n, h}

P = Field that the curve is define over

G = Generator point

a, b = Values define the curve

h = Co- factor

n = Prime order of G

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 received the key and from 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 = $x = G^a mod P$	Key generated = $y = G^b mod P$
Exchange of generated keys takes place	
Key received = y	key received = x
Generated Secret Key = $k_a = y^a mod P$	Generated Secret Key = $k_b = x^b mod P$
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 = 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.
```

Implementation:

```
/* 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 primitve 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 b
    b = 3; // b is the chosen private key
    printf("The private key b for Bob : %lld\n\n", b);
    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

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

    return 0;
}
```

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

Secret key for the Alice is : 9
Secret Key for the Bob is : 9
```

This article is contributed by **Souvik Nandi**. If you like GeeksforGeeks and would like to 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.

Recommended Posts:

RSA Algorithm in Cryptography
Computer Network | Leaky bucket algorithm
Computer Network | HMAC Algorithm
TCP Server-Client implementation in C
Probabilistic shortest path routing algorithm for optical networks
RSA Algorithm using Multiple Precision Arithmetic Library
UDP Server-Client implementation in C
Computer Network | RC4 Encryption Algorithm
Back-off Algorithm for CSMA/CD
Computer Network | Algorithm for Dynamic Time out timer Calculation