Baby step - Giant step algorithm

Milan Pavlović, LogN team

This is another blog about topics that we at LogN team meet while doing research in area of Zero Knowladge. Point of interest in this blog is Baby step - Giant step algorithm.

Keywords: discret log, algorithm, cryptography, Baby step - Giant step, LogN

TABLE OF CONTENTS

- Intro: Elliptic curves and DLP
- Algorithm
- Example
- Conclusion
- Code
- Resources

INTRO: ELLIPTIC CURVES AND DLP

Elliptic curve $E(\mathbb{F})$ over field $\mathbb{F}$ have structure of additive group, more specific abelian group. Details of that structure demands whole another blog, interested reader is refered to references on the end of the blog.

Problem that lies in basis of elliptic curve cryptography is:

Given two points, say P and Q in subgroup G of elliptic curve group $E(\mathbb{F}_q)$ over finite field $\mathbb{F}_q$, find integer k such that $P = [k]Q$ (notation widely accepted for additon on elliptic curve).

Problem cited above is called elliptic curve discrete logarithm problem (ECDLP) and there is no known polynomial time algorithm for solving that can run on a classical computer. It is ”elliptic” because we are doing calculations on elliptic curve, ”discret” because coordinates of points are integers from finite field and ”logarithm” because it is analogous to classical mathematical logarithm.

Discrete logarithm problem(DLP) is also known in other cryptosystems such as the Digital Signature Algorithm (DSA), the Diffie-Hellman key exchange (DH) and the ElGamal algorithm. There is reason why they share the name, they represent similar problems. Difference between ECDLP and DLP is that in ECPLD we work with additive group of ellpitic curve while in DLP one works with multiplicative group. The DLP problem can be stated as:

Given y and g from group and integer p, find x such that $y = g^x \pmod{p}$.

In this blog we'll present how one can solve DLP for prime p, similar ideas then can be applied for ECDLP. We choose this path for clarity of presentation, to avoid dealing with elliptic curves.

What makes elliptic curves useful in cryptography is that, as of today, the discrete logarithm problem for elliptic curves seems to be ”harder” if compared to other similar problems used in cryptography (intuitively, the reason is complexity of elliptic curve). This implies that we need fewer bits to achieve the same level of security as with other cryptosystems.

ALGORITHM

We'll now describe algorithm used to solve DLP, which is, due to Daniel Shanks, called Baby step - Giant step. This algorithm can be applied to any finite cyclic abelian group. Depending of use case some modifications are posible.

Asume we have public cyclic group $G = \langle g \rangle$ of prime order p. Given $y \in G$, we are asked to find value $x \in \mathbb{Z}_p$ such that $y = g^x \pmod{p}$.

Idea behind algorithm is often encountered idea of divide and conquare. We first calculate $k = [\sqrt{p}] + 1$. Then we write:

$$x = x_0 + x_1 \cdot k,$$

where $0 \leq x_0,\ x_1 \leq p$. To continue, we calculate baby steps:

$$g_i = g^i,\ 0 \leq i < k.$$

The values g_i are then stored in array X. To compute and store these values requires $\mathscr{O}(\sqrt{p})$ time and space. We then proceed to Giant steps:

$$y_j = y \cdot g^{-jk},\ 0 \leq j < k.$$

After calculating each y_j, we try to find that value in array X. If such match is found, say $X[i] = y_j$, we have solution to our DLP and that solution is:

$$x = i + j \cdot k \ (\text{mod } p).$$

We can analyze last assertion a bit. Match $X[i] = y_j$ is equivalent to:

$$g^i = y \cdot g^{-jk}$$

$$g^{i+j \cdot k} = y \ (\text{mod } p).$$

From last equation we read solution to our discrete logarithm problem, $x = i + j \cdot k \ (\text{mod } p)$. Notice that Giant step requires at most $\mathscr{O}(\sqrt{p})$ time. Hence, the overall time and space complexity of the Baby step - Giant step algorithm is $\mathscr{O}(\sqrt{p})$. This means that if we want our problem to bi 2^{128} difficult, we need to take prime of order 2^{256}.

EXAMPLE

As an ilustration let's take finite field $\mathbb{F}_{509}$ and take a closer look on it's multiplicative subgroup G of order 127 generated by element $g = 17$. Further on, integer $y = 438$ is given and we want to find x such that:

$$438 = 17^x \ (\text{mod } 509),$$

i.e. our task is to solve DLP.

Following presented algorithm, we first calculate ceiling $k = [\sqrt{509}] + 1 = 23$. Then, we proceed to Baby steps:

$$g_i = 17^i,\ 0 \leq i < k.$$

Values g_i are stored in array X which is given in table below in format $i : 17^i \ (\text{mod } 509)$.

0: 1	6: 280	12: 14	18: 357
1: 17	7: 179	13: 238	19: 470
2: 289	8: 498	14: 483	20: 355
3: 332	9: 322	15: 67	21: 436
4: 45	10: 384	16: 121	22: 286
5: 256	11: 420	17: 21	

Table 1. Array X in format $i : 17^i \ (\text{mod } 509)$.

Next step is Giant step. We calculate

$$y_j = 438 \cdot g^{-k \cdot j},\ 0 \leq j < k,$$

and after each calculation try to find same value in table above. In this example, calculations gives us:

$y_0 = 438$, which is not value from table;

$y_1 = 199$, which also is not found in table;

$y_2 = 238$ is in table. We memorize index of y, that is $j = 2$.

Value 238 is calculated in Baby step for $i = 13$, so solution to our DLP is $x = 13 + 2 \cdot 23 = 59$. You can check that $17^{59} = 438 \ (\text{mod } 509)$ indeed.

Optimization can be made if one knows that g is in some subgroup of $\mathbb{F}_p$. As we know here that g is in subgroup G of order 127, order of that subgroup can be used for calculating ceiling $k = [\sqrt{127}] + 1 = 12$ instead of order 509 of whole group which yields $k = [\sqrt{509}] + 1 = 23$. This optimization has its own cost, we have to determine order of g.

CONCLUSION

There are a lot applications of Baby step - Giant step algorithm, and some modifications. One limitation of algorithm is that it has $\mathcal{O}(\sqrt{p})$ space complexity, beside $\mathcal{O}(\sqrt{p})$ time complexity. It's a big drawback and reason why DLP is hard for todays computers. One optimization in that direction is Pollard's Rho algorithm. Other generalization is Pohlig-Hellman algorithm which is used when integer p in DLP is not prime. This algorithm reduces the DLP from a group of composite order to subgroups of prime order, then solve DLP in each subgroup. Chinese Remainder Theorem is used to reconstruct solution of original problem. One of important conclusions from Pohlig-Hellman algorithm is that DLP in group G is as hard as it is DLP in its largest subgroup of prime order (which dominates in time and space complexity). This idea lies in known Pohlig Hellman small subgroup attacks.

CODE

Small Python script for our example (calculations in our example are tested in SageMath too):

```
from math import sqrt

def babyGiant(g, y, p, option = False):
    '''
    Returns x s.t. y = g^x mod p for a prime p.
    optional argument for printing intermediate steps
    '''
    # calculating ceiling
    k = round(sqrt(p - 1))

    X = [None] * k

    # Baby steps
    for i in range(k):
        X[i] = pow(g, i, p)

    #Calculating inverse using Fermat's little theorem
    inv = pow(g, k * (p - 2), p)

    if option:
        print("X: ")
        for i in range(k): print(i, ':', X[i])
        Y = [None] * k

    # Giant step, search for match in the list
    for j in range(k):
        tmp = (y * pow(inv, j, p)) % p
        if option: Y[j] = tmp

        if tmp in X:
            if option: print("Y: ", Y[:j+1])
            return X.index(tmp) + j * k
    # Solution not found
    return None

print(babyGiant(17, 438, 509, True))
```

RESOURCES

- Cryptography: An Introduction, Nigel Smart
- Rational Points on Elliptic Curves, Joseph H. Silverman and John T. Tate
- Pairings for beginners, Craig Costello
- https://andrea.corbellini.name
- https://www.sagemath.org