Discrete Logarithm is Easy in Modulo Fermat Prime Groups

Anindya Shankar Bhandari

Indian Institute of Technology, Kharagpur

***Abstract*:- This paper deals with proposing an algorithm that can solve DLP in polynomial time in certain groups. It is an extension to the general algorithm proposed by Douglas Long and AviWigderson in their paper “How Discreet is the Discrete Log?", except that the main algorithm proposed in this paper works without needing to find any square roots. The first algorithm proposed is a general purpose algorithm, and solves DLP in the general case, but no method has been located to be able to employ that. The second algorithm solves DLP in specific groups, particularly modulo Fermat Prime or Psuedo Fermat Prime groups. This has been discussed in detail, and can be used readily. It may also be used to reveal a certain number of bits of the exponent in other groups. The exponent-construction based algorithm may also find uses in solving DLP in other groups as well, by checking the coprimality of the exponent (and various functions of the exponent) against the factors of N-1. All exponentiations done in relation to the generator are modular. This does not apply to exponentiation of 2 to define the group size, or the complexity analysis.**

***Keywords:* Discrete Logarithm; DLP; Quadratic Residue; Finite Field; Number Theory; Fermat Prime.**

I. INTRODUCTION

An integer x that solves the equation g^x=a, where both g and a are elements of a finite group, is referred to as a discrete logarithm. It is the analogue of an ordinary logarithm in finite groups. There is no known way to efficiently compute the solution of the Discrete Logarithm Problem.

In this paper we deal with finite cyclic groups that are modulo a prime, and the generator g is chosen to be a primitive root modulo that prime.

A primitive root modulo a prime is an element that generates a finite cyclic group (by exponentiation of the generator, in multiplicative groups) in which all elements are distinct.

II. FERMAT PRIMES

Fermat Primes are primes of the form 2^{2^n}+1.

This paper discusses an algorithm that works for primes of the form 2^m+1 (and some other forms, this is discussed later). However, it has been shown that if 2^m+1 is a prime, then m must be a power of 2. So far, the only known Fermat Primes are 3, 5, 17, 257, 65537. It is speculated that no new Fermat Primes will be found with current computational hardware limitations.

Numbers of this form which are not prime are considered “Pseudo Fermat Primes" as they satisfy the properties of the prime number in Fermat's Little Theorem. Hence these numbers may also be used as a finite group limit for DLP.

III. SQUARE ROOTS IN FINITE FIELDS AND THEIR RELATION TO THE DLP

Let us assume there is a way to check if any number modulo N (a prime) is even or odd, and there is a way to perform an integer division on that number.

Therefore, if x is even, x' takes the value :$\frac{x}{2}$

And, if x is odd, x' takes the value $\frac{x-1}{2}$.

A recursive formulation to find x in O(log{N}) time is discussed below :

f(g^x, obj K, integeri)

```
Step 1: Base Condition. if$g^x$ = 1, reconstruct x with the operation history stored in K. Reconstruct(K,i), exit function
Step 2: if x is even, do steps 3-4
Step 3: k[i+1]=even
Step 4: call f($g^{\frac{x}{2}}$, k,i+1)
Step 5: if x is odd, do steps 6-7
Step 6: k[i+1]=odd
Step 7: callf($g^{\frac{x-1}{2}}$,k,i+1)
```

The time complexity of this algorithm is of the order of logN times the complexity of even/odd checking.

One way to check if x is even or odd is to raiseg^x to the power $\frac{N-1}{2}$.

For x=even, this gives the value 1.

For x=odd, this gives the value N-1.

However the division of x by 2 is not directly possible. One solution would have been to take the square root of g^xor g^{x-1}modulo N (if x is a quadratic residue). However, this would give us both $g^{x'}$ and $-g^{x'}$, that is, basically, $g^{x'}$and $g^{\frac{N-1}{2}+x'}$,where x' is defined as above, for even/odd x. **If the problem of finding the “best square root" could be solved in polynomial time, DLP would be solved.**

IV. THE SPECIAL CASE ALGORITHM

Let us assume that N is of such a form that we are able to raise g to the power $\frac{N-1}{2^k}$.
till x has been found, hence $x < 2^k$
The algorithm is described as follows. Initial values of k and a are 1 and 0 respectively. The value $g^x$is available.

f(k,a)
Step 1: Base condition. Check if $g^{x-a}=1$. If yes, return a.
Step 2: Raise $g^{x-a}$to the power $\frac{N-1}{2^k}$and check if it is equal to one. If yes, return f(k+1,a)
Step 3: If not equal to one, return f(k,a+2^{k-1}).

Values of $\frac{N-1}{2^k}$ and 2^kupto a certain recursive step are stored in memory, to reduce any additional complexity arising due to the calculation of these.

4.1 Complexity Analysis and Correctness

The aim of the algorithm is to perform integer division without actually dividing. There are a maximum of 2 comparisons required at each step. Including the modular exponentiation, the algorithm has a running time of

$$2\sum_{k=1}^{log x} log\frac{N-1}{2^k}.$$

Or, in the worst case, O(log^2N).
The integer division works like this, if x' = $\frac{x-1}{2}$ is even, x"= $\frac{x}{2}$= $\frac{x-1}{2^2}$. If x' is odd, x"= $\frac{x-1}{2}$= $\frac{x-3}{2^2}$. Hence 'a' gets added with 1 multiplied by the denominator.

4.2 Groups

This algorithm works in groups modulo primes of the form 2^n+1. However, it also works in groups modulo primes of the form $2^n{}_1+2^n{}_2+2^n{}_3$, and so on, +1, provided x <= the lowest power of 2 in the summation (excluding 1). It may be possible to create an algorithm that reduces x to a neighbourhood within that lowest power of 2 in polynomial time.
The interesting thing is, if the group size is of the form $2^n{}_1+2^n{}_2+2^n{}_3+...+2^n{}_k+1$, where $n_k$is the lowest power of the exponent, and x >$2^n{}_k$,the algorithm may still be used to reveal n_k bits of the exponent x, thereby reducing the complexity of solving DLP for that specific group by $2^n{}_k$.

4.3 Comments

The algorithm works without actually performing any kind of square root operation. As opposed to Long and Wigderson's work, where they consulted an oracle for the correct square root (incorrectly referred to as the principal square root - the principal square root returns the root of the quadratic residue that is also a quadratic residue), this algorithm works by simply “constructing" the appropriate exponent that corresponds to the roots of x or x-1.

V. CONCLUSION

We proposed an algorithm to efficiently compute the discrete logarithm in groups modulo a Fermat Prime, and certain other prime forms, subject to certain limitations. This same algorithm may be used in those other groups to reveal a number of bits of the exponent. We also proposed a general case algorithm, but this is not usable efficiently as there are 2 square roots modulo a prime, and so far, it is not possible to know which is the positive square root, given that x of $g^x$is unknown. **This construction algorithm (which does not require one to find any square roots) may also be used to check the coprimality of the exponent x (and neighboring values of x, and functions of x) against the factors of N-1.**It is therefore our belief that this algorithm can be useful in solutions to DLP in various other groups as well.

REFERENCES

The following references have been used in preparation of this paper.

[1]. Douglas L. Long, AviWigderson (1983). “How discreet is the discrete log?", The 15th annual ACM symposium on Theory of computing.
[2]. Theory of Quadratic residues - Wikipedia. https://en.wikipedia.org/wiki/Quadratic_residue
[3]. Discrete logarithm - Wikipedia. https://en.wikipedia.org/wiki/Discrete_logarithm