Baby-Step/Giant Step DL Algorithm

William Gasarch

July 14, 2015

1 A Bad Way to Computer Discrete Log

Definition 1.1. The *Discrete Log Problem* is as follows. Let p be a prime and g be a generator. These are considered parameters. Given $y \in \{1, \ldots, p-1\}$ find x such that $y = g^x$.

Throughout this paper p is a prime and g is a generator for it.
Here is an algorithm for DL that is slow.

1. Input(y)
2. For $x = 0$ to $p - 1$
   (a) Compute g^x.
   (b) If $g^x = y$ then output(x) and STOP

Note that in the worst case this could take roughly p steps. But worse than that—its average case also is not very good (we will not go into this). What we mean is that in MANY cases it will take a lot of time.

2 Fast(er) Algorithm for Calculating Discrete Logs: the Baby-Step / Giant-Step Algorithm

Can we do better than p steps? How much better? Recall that we thing of p as large and $\log p$ as small. Hence if DL could be solved in $\log p$ steps even $(\log p)^2$ steps or even $(\log p)^{10}$ steps that would be fast. Alas we cannot do that. However, we can do DL in roughly $\sqrt{p}$ steps.

First we do some informal reasoning. Define $m = \mathsf{Floor}(\sqrt{p-1})$. Thus whenever we do something m times, we are doing it roughly $\sqrt{p}$ times.

Given y we want to find x such that $y = g^x$.

KEY IDEA ONE: Think of dividing x by m. Thus $x = qm + r$. Since we divided by m we know that $0 \leq r \leq m$. Since $m \times m$ is roughly p we can assume $0 \leq x \leq m$.

KEY IDEA TWO: Lets say we GUESS what q is. If we are CORRECT then there is some r such that

$y = g^{qm+r}$

$y = g^{qm} g^r$

$g^r = y \times g^{-qm}$

AND recall that $0 \leq r \leq m$. So if we guess q correctly then $y \times g^{-qm}$ has to itself have discrete log $\leq m$.

KEY IDEA THREE: Prepossess! Before doing any discrete logs have tables of powers of g that are helpful. Then we can check if some numbers has DL $\leq m$. Can't prepossess too much since that would take too much time.

We will now do the PREPOSSESS and ALGORITHM.

PREPROCESSING

1. For $i = 0$ to m computer g_i. Put them in a table that looks like this (I made the numbers up and so they are NOT TRUE. We are assuming $p = 7$ for illustration.)

i	g^i
1	6
2	4
3	2
4	5
5	3
6	1

Now SORT the table on the SECOND entry. Hence we have, in our example,

i	g^i
6	1
3	2
5	3
2	4
4	5
1	6

2. For $i = 0$ to m compute g^{-im}. (Actually we need g^{-1} for this.) Put in a table. Example with made up numbers with 7. So here $m = 3$.

i	g^{-3i}
1	3
2	5
3	4
4	1
5	2
6	6

No need to sort.

SO, we have both of these tables.

ALGORITHMS

1. Input(y) (We seek x such that $y = g^x$. We think of x as $qm + r$ with $0 \leq q, r \leq m$ and hence we seek q, r.)

2. For $q = 0$ to m we TRY to see if q is a good guess for q.

   (Comment- if q is correct then $y = g^{qm+r} = g^{rm}g^r$, so $g \times g^{-qm} = q^r$)

   Find g^{-qm} on the second table.

   Compute $y \times g^{-qm}$. KEY- Look for this in the first table (sorted version). Use binary search (so take roughly $\log m$ steps).

   IF find it then find the r such that $g^r = g^{-qm}$. OUTPUT $x = qm + r$.

   IF do not find it then FINE, just go to next value of q.

How long does this take. The preprocessing takes roughly m steps. But you may not want to really count that since you do it once and may use the data over and over again.

The algorithm also takes roughly m iterations, each one takes $\log p$ steps. So this takes $m\sqrt{p}$ steps.

Since that m is roughly $\sqrt{p}$. So the entire algorithm takes $\sqrt{(}p) \log p$ steps.

3 What of it?

So DL can be solved in $\sqrt{p}$ steps. This is still not that fast. However, its far faster than we thought. Lets say that 2^{100} is too big for a computer to do that many steps.

OLD MENTALITY: Need p such that $\log p$ is small and p is large. Take $p = 2^{100}$.

NEW MENTALITY: Need p such that $\log p$ is small and $\sqrt{p}$ is large. Take $p = 2^{200}$.

This is a warning. Clever math has not yet made us think that DL is easy (and hence Diffie Helman is breakable but this clever math has made us increase our parameter.

State of the art: So far better algorithms for TIME are not known. However, note that our algorithm takes $\sqrt{p}$ space which is alot. There are algorithms that take $\sqrt{p}$ time and much less space then the one presented.