Sketch of a 653 bit safe prime discrete logarithm problem

March 2, 2019

In 2016, Kleinjung finished the computation of a discrete logarithm in a 768-bit prime field with $p = [2^{766}\pi] + 62762$. Such a p is called a safe prime as $\frac{p-1}{2}$ is prime too. A safe prime makes the discrete logarithm hard because there is no smaller subgroup where we can transit the computation.The result took the equivalent of about 5300 core years on a single core of a 2.2GHz Xeon E5-2660 processor and it is a record for computing prime field discrete logarithms. In 2017, JPNE[17] finished the discrete logarithm computation of a 1024-bit prime in its 160-bit subgroup. 400 core-years were spent in this computation. Except for the different software used between these two computations, there are two main reasons for the efficiency of the latter. Firstly, the sieve step can be accelerated since there are more relations found due to the property of the special number field sieve polynomials. Secondly, since the 768-bit prime is safe, the linear algebra must be performed modulo $\frac{p-1}{2}$. It is much more expensive than the 160-bit linear algebra used for the 1024-bit prime. Also, as the general number field sieve has worse smoothness probabilities, the matrix in the linear algebra of the 768-bit prime is a bit larger thus increase the cost.

In this paper, we choose $p = 22667110984426410454539321563351005059298050926003323907675540112769012557529111542664501825859542434399689453609566803813691163927790935258104921196863722947966948283866110442812635122066048059643$ which is a 653-bit safe prime. Let $g = 5$ be the primitive element of the prime field F_p and the base of the discrete logarithm. The target chosen is $t = 2030050705925408245552747951968514865310881176818236467269758838646428343677065394120477653293862344579580485440879323173591611705705375608391817202160788513498358451250900864760571670763377 58$. After about 318

hours calendar time computation or equivalently 1.74 core years on a HP-workstation with 48 cores of 2.2GHz Intel Xeon E5-2946v4 processor, we got the result $log_g t = 11711391724874890704542871336515356421985211880470075302$
$4944524528637796710431135053982960728989194239020461385299981487744704863$
$7883144655337721464521946063017119883985375869680331050660426050237 4$

The main point of our paper is to give an easy and practical way to detect if there is a special number field sieve. Our idea is inspired by the lattice used in Lercier and Joux's two algebraic sides polynomial selection(JL[03]) and a computation using Kleinjung's polynomial selection method in the cado-nfs tool. It is an old problem to detect a trapdoor prime in Gordon[93] and was raised again in JPNE[17]. Their main idea is to construct a trapdoor so the way they detect a SNFS method is not so practical compared to ours. We shall show their methods briefly in the following:
Given a prime p , they choose the degree k of the polynomial f and search for a f with small coefficients (use a bound for the coefficients). Then they will try to find two integers X and Y which satisfy $Y^k f(\frac{X}{Y}) \equiv 0(mod\ p)$. They show such a pair (X,Y) can be found by lattice reduction as a short vector once f is determined. Then homogeneous polynomials of degree one and k with small coefficients can be used for the special number field sieve.
While our method to employ a special number field sieve is different. Given a prime p, we don't search or determine f first. Actully, we also choose the degree k of f but we decide its root $modp$ first. We first pick an interger m which is around $[p^{\frac{1}{k}}]$. We want to get a polynomial f with small coefficients and degree k. We construct a $k+1$ dimension lattice:

$$A = \begin{bmatrix} p & 0 & 0 & \cdots & 0 \\ -m & 1 & 0 & \cdots & 0 \\ -m^2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -m^k & 0 & 0 & \cdots & 1 \end{bmatrix}$$

As k is always from3 to 7, the dimension of A is low enough to determine its shortest vector by BKZ in the tool NTL. We employ lattice reduction to A by NTL and the first line of A will be the coefficients of the polynomial

f. Actually Joux used this method to get a degree k polynomial from a degree $k+1$ polynomial. Since we have decided the root m we just get the algebraic polynomial f and the rational polynomial $g = x - m$
In our example we choose $k = 5$ and $m = 186672023076503556998251248696793510245 0$, then we construct the lattice A and get $f(x) = x^5 + x^4 - 3x^3 - 13x^2 + 7x - 7$ and $g(x) = x - 186672023076503556998251248696793510245 0$

1 conclusion

We used this polynomial pair in the computation of cado-nfs and it performs much better than the method of Kleinjung's. This is also a variant of the Base-m method as Kleinjung[06] introduces a vector to control the third term rather than translate m. But it is not enough to shorten athe time when the given prime p is too large.
Although the probability that a given prime is vulnerable to SNFS method is zero but if it really happens it is 1. So I think we can put an easy search course into the polynomial selection phase as an ad-hoc attack.

2 Reference

0. Computation of a 768-bit prime field discrete logarithm
1.JPNE[17] : A Kilobit Hidden SNFS Discrete Logarithm Computation
2.Kleinjung[06] : On Polynomial Selection for the General Number Field Sieve
3.JL[03]: Improvements to the general number field sieve for discrete logarithms in prime fields.
4. Gordon[93]: Designing and detecting trapdoors for discrete log cryptosystems
other : The development of the number field sieve