The Special Number Field Sieve in $\mathbb{F}_{p^n}$

Application to Pairing-Friendly Constructions

Antoine Joux[1,2,3] and Cécile Pierrot[2,4]

[1]CryptoExperts
[2]Université de Versailles Saint-Quentin-en-Yvelines, Laboratoire PRISM,
45 avenue des États-Unis, F-78035 Versailles Cedex, France

[3]Antoine.Joux@m4x.org
[4]Cecile.Pierrot@prism.uvsq.fr

September 9, 2013

Abstract In this paper, we study the discrete logarithm problem in finite fields related to pairing-based curves. We start with a precise analysis of the state-of-the-art algorithms for computing discrete logarithms that are suitable for finite fields related to pairing-friendly constructions. To improve upon these algorithms, we extend the Special Number Field Sieve to compute discrete logarithms in $\mathbb{F}_{p^n}$, where p has an adequate sparse representation. Our improved algorithm works for the whole range of applicability of the Number Field Sieve.

1 Introduction

Since its introduction, pairing-based cryptography has permitted the development of many cryptographic schemes, including identity-based cryptographic primitives [BF03, SK03, CC03, Pat02], short signature schemes [BLS04], or one-round three-way key exchange [Jou04]. Some of these schemes have already been deployed in the marketplace.

One very important challenge necessary for practical pairing-based cryptography is to construct pairing-friendly curves suitable for efficient asymmetric schemes offering a high security level and to estimate precisely their concrete security. This is not a simple task, since it requires to construct a pairing-based curve, while balancing the complexities of the various discrete logarithm algorithms that can be used. This challenge has been studied in many articles such as [BLS03, KM05]. To evaluate the security of a given construction, the traditional approach is to balance the complexity of square-root algorithms for computing discrete logarithms in the relevant subgroup of the elliptic curve and an estimate of the complexity of the Number Field Sieve (NFS) algorithm in the finite field where the pairing takes its values. The complexity of solving the discrete logarithm problem in the finite field in this context is usually estimated by using keysize tables such as [Nat03, LV01].

This approach makes an *implicit assumption,* namely it considers that the complexity of NFS in the finite field is close to the complexity of factoring an integer of the same size. As far as we know, this implicit assumption has not been checked in the relevant literature.

Our goals in this paper is twofold. First, we show that for current parameters of high-security pairing-based cryptography, the implicit assumption is incorrect and that the state-of-the-art algorithm for discrete logarithms in this case is the High-Degree variant of the Number Field Sieve introduced in [JLSV06], whose complexity is higher than the complexity of factoring. Second, we revisit discrete logarithm algorithms and show that, thanks to the specific form of the characteristic of the field of definition of pairing-friendly curves that appears with classical constructions, such as Barreto-Naehrig [BN05], it is possible to devise improved algorithms by generalizing the Special Number Field Sieve (SNFS) [Gor93, Sch08]. In truth, we go beyond this goal and present SNFS variations for the whole range of finite fields covered by the NFS, assuming that the characteristic of the field admits an adequate sparse representation.

This paper is organized as follows: in Sections 2 and 3 we make a short refresher on tools and on the Number Field Sieve used in the medium prime case. We explain in Section 4 the state-of-the-art algorithm for discrete logarithms for pairing-based cryptography. In Section 5 we develop our variation on the special index calculus algorithm for $\mathbb{F}_{p^n}$ involved in pairing-friendly constructions. The Section 6 gives finally a precise heuristic analysis of the algorithm.

2 Tools and Notations

When dealing with index calculus algorithms, the complexity are usually expressed using the following notation:

$$L_q(\alpha, c) = \exp\left((c + o(1))(\log q)^{\alpha}(\log\log q)^{1-\alpha}\right)$$

where α and c are constants such that $0 < \alpha < 1$ and $c > 0$ and log denotes natural logarithm. The notation $L_q(\alpha)$ is also used when the constant c is not explicitly specified. These functions arise from the probability for values to be smooth. More precisely, it comes from the well-known theorem of Canfield, Erdös and Pomerance. Let us introduce the quantity $\psi(x, y)$ to denote the number of positive numbers up to x which are y-smooth. Then $\log(\psi(x, y)/x)$ is close to:

$$-\frac{\log x}{\log y}\left(\log\left(\frac{\log x}{\log y}\right)\right).$$

In particular, when both x and y are given as L_q expressions, we have the following theorem.

Theorem 2.1 (Canfield, Erdös, Pomerance)**.** *Let $C \subset \mathbb{R}^4$ be a compact set such that for all $(r, s, \gamma, \delta) \in C$ one has $\gamma > 0$, $\delta > 0$ and $0 < s < r < 1$. Let $\mathcal{P}$ denote the probability that a random positive integer below $L_q(r, \gamma)$ splits into primes lower than $L_q(s, \delta)$. Then we have:*

$$\mathcal{P} = L_q(r - s, -\gamma(r - s)/\delta + o(1))$$

for $q \to \infty$, uniformly for (r, s, γ, δ) in C.

As often the case in articles studying index calculus algorithms, we also use the *heuristic generalization* of Theorem 2.1 to the probability of smoothness of integers which are not selected uniformly and to pairs of integers which are not independent.

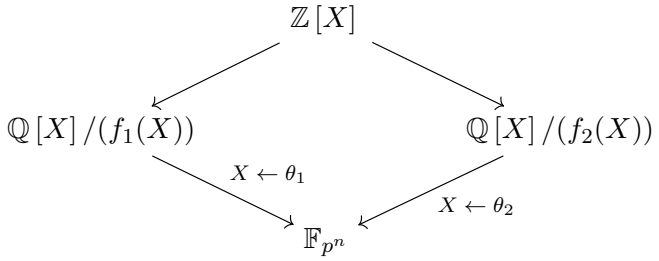

Figure 1: Commutative diagram for the algorithm of [JLSV06]

3 A Short Refresher on Discrete Logarithms in the Medium Prime Case

Thoughout the rest of the paper, Q denotes the size of the finite field being considered, *i.e.* $Q = p^n$.

3.1 The $p = L_Q(l_p, c_p)$ Case, with $1/3 \leqslant l_p < 2/3$

We first recall the Number Field Sieve variant proposed in [JLSV06] in the case where $p = L_Q(l_p, c_p)$, with $1/3 < l_p < 2/3$. We extend afterwards the algorithm to the configuration $p = L_Q(1/3, c_p)$.

3.1.1 The $l_p \neq 1/3$ Case

Setup

General setting. In order to compute discrete logarithms in $\mathbb{F}_{p^n}$, a degree n extension of the base field $\mathbb{F}_p$, we start by choosing two polynomials f_1 and f_2 in $\mathbb{Z}[X]$ with a common root in F_{p^n}. In other words, we choose f_1 and f_2 such that the greatest common divisor of these two polynomials has an irreducible factor of degree n in $\mathbb{F}_p$. As a consequence, we can draw the commutative diagram in Figure 1.

Let $\mathbb{Q}[\theta_1]$ denote $\mathbb{Q}[X]/(f_1(X))$ and $\mathbb{Q}[\theta_2]$ denote $\mathbb{Q}[X]/(f_2(X))$, the two number fields defined by f_1 and f_2, *i.e.* θ_1 and θ_2 are roots of these polynomials in $\mathbb{C}$.

Choice of polynomials. In [JLSV06], f_1 is chosen as a degree n polynomial, with small coefficients and irreducible over $\mathbb{F}_p$, while f_2 is defined as the polynomial $f_1 + p$. To balance the size of the norms computed during the algorithm, another approach is also mentioned. This variant uses continued fractions and involves changing the polynomial selection such that the coefficients of both polynomials are of the same size. One possibility is to choose f_1 such that at least one of its coefficients, let us say c, is of the order of $\sqrt{p}$. More precisely, such that $f_1 = g + c \cdot h$ where g and h are polynomials with small coefficients. Since we can write $c \equiv a/b \pmod{p}$ with a and b also of the order of $\sqrt{p}$, we define $f_2 \equiv bf_1 \pmod{p}$. The coefficients of f_2 are $O(\sqrt{p})$ instead of $O(p)$. The key contribution of our variation of the SNFS in Section 5 is to reduce the size of the coefficients that appear in f_1 and f_2.

Sieving

We then denote by $t-1$ the degree[1] of the polynomials which we are going to sieve on and the two following bounds: A a sieve limit and B a smoothness bound. We consider all t-uples of the form $(a_0, \cdots, a_{t-1})$ such that the norms of the $a_0 + \cdots + a_{t-1}\theta_1^{t-1}$ and $a_0 + \cdots + a_{t-1}\theta_2^{t-1}$ are both B-smooth. After some post-processing described in [JLSV06], each such t-uple yields a linear equation between "logarithms of ideals" coming from both number fields and belonging to the smoothness basis.

Linear Algebra

Once the sieving phase is complete, we solve the resulting sparse system of equations modulo $p^n - 1$, the cardinality of $\mathbb{F}_{p^n}^*$, and recover logarithms of ideals in the smoothness basis. To be more precise, the linear algebra is done modulo a large factor of this cardinality, while small prime factors are considered separately, using a combination of Pollard's rho and Pohlig-Hellman algorithm.

Individual Discrete Logarithms

Once the sieving and the linear algebra phases have been performed, we know the logarithms of all the ideals in the smoothness basis. The important phase that remains is the individual discrete logarithms phase which allows to compute the discrete logarithm of an arbitrary element in the finite field. The approach proposed in [JLSV06] is based on a "special-**q**" descent. In a nutshell, in order to compute the discrete logarithm of an arbitrary element y of $\mathbb{F}_{p^n}$, represented as $\mathbb{F}_p[X]/(f_1(X))$, we search for a multiplicative relation of the form $z = y^i X^j$ for some $i, j \in \mathbb{N}$. We expect z to satisfy the two properties: if $B' = L_Q(2/3, 1/3^{1/3})$ and $\bar{z}$ denotes the lift of z to the number field $\mathbb{Q}[\theta_1]$, the norm of $\bar{z}$ should be B'-smooth and squarefree. The second condition implies that only degree one prime ideals will appear in the factorization of $(\bar{z})$. This is required since only logarithms of such ideals are computed during the sieving and linear algebra phases. After finding such a z, we factor the principal ideal generated by $\bar{z}$ into degree one prime ideals of small norm. Some of them are not in the factor base (those whose norm is smaller than B' but bigger than B). To compute the logarithm of such an ideal $\mathbf{q}$ we start a "special-**q**" descent, progressively lowering the bound B' until it reaches B. We finally backtrack to recover the logarithm of $\bar{z}$ and consequently the logarithm of y.

3.1.2 The $p = L_Q(1/3, c_p)$ Case

Again, we sieve on polynomials of degree $t-1$ and follow the phases described above. However, the analysis of [JLSV06] no longer applies directly and we need to refine the bound on norms to extend the analysis to this case.

A short analysis of the extended NFS

Let us recall that the parameters of the algorithm taking part in the analysis are the extension degree n, the smoothness bound B, the degree $t-1$ of the elements we are sieving over and the bound A on the coefficients of these elements. We assume that we

[1] While this notation might seems awkward, it is in fact quite convenient, because a polynomial of degree $t-1$ gives dimension t to the sieving space.

can write:

$$n = \frac{1}{c_p}\left(\frac{\log Q}{\log\log Q}\right)^{2/3}, \quad t = \frac{c_t}{c_p}\left(\frac{\log Q}{\log\log Q}\right)^{1/3}, \quad A^t = L_Q(1/3, c_a c_t), \quad B = L_Q(1/3, c_b)$$

where c_a, c_b and c_t will be determined later on.

The difference between the case $p = L_Q(1/3, c_p)$ and the case $p = L_Q(l_p, c_p)$ with $1/3 < l_p < 2/3$ appears when computing the bound on the norms of polynomials. More precisely, for each polynomial involved in the sieving, the two norms are easily bounded by $n^t t^n A^n$ and $n^t t^n A^n p^t$ respectively on the $\mathbb{Q}[\theta_1]$ and on the $\mathbb{Q}[\theta_2]$ side. When $l_p > 1/3$, the $n^t t^n$ term is negligible compared to A^n and p^t and vanishes from the complexity analysis. However, due to the size of t and n in the $l_p = 1/3$ case, the t^n term is no longer negligible. This leads to an extra term in the product of the norms and yields a $L_Q(1/3)$ complexity with a constant higher than $(128/9)^{1/3}$.

We now improve the bound on these two norms in order to restore a complexity of $L_Q(1/3, (128/9)^{1/3})$ in this extended case of NFS.

First, each norm can be expressed as the determinant of the Sylvester matrix $M(f_i, h)$ of f_i and h, with $i = 1$ or $i = 2$. Let $m_{j,k}$ represents the coefficient at the j-th row and k-th column of $M(f_i, h)$. Using the formula:

$$\det M(f_i, h) = \sum_{\sigma_j \in \mathrm{S}_{n+t}} \mathrm{sign}(\sigma_j) \prod_{k=1}^{n+t} m_{k,\sigma_j(k)}, \tag{1}$$

we remark that the two norms are bounded by ΘA^n and $\Theta A^n p^t$ respectively on the $\mathbb{Q}[\theta_1]$ and on the $\mathbb{Q}[\theta_2]$ side, where Θ is the number of permutations of S_{n+t} leading to a non zero product in (1). Kalkbrener gives a majoration of Θ in the second theorem of [Kal97]. With our notations this bound becomes: $\Theta \leqslant \binom{n+t}{n} \cdot \binom{n+t-1}{t}$. Because of the following inequalities:

$$\begin{aligned}
\tbinom{n+t}{n} \cdot \tbinom{n+t-1}{t} &= \frac{n}{n+t}\left(\frac{(n+t)!}{n!t!}\right)^2 \\
&\leq \frac{n}{n+t}\left(\frac{(n+1)\cdots(n+t)}{t!}\right)^2 \\
&\leq \frac{n}{n+t}\left(\prod_{i=1}^{t}\frac{(n+i)}{i}\right)^2 \\
&\leq \frac{n}{n+t}\prod_{i=1}^{t}\left(\frac{n}{i}+1\right)^2
\end{aligned}$$

we obtain that $\Theta \leqslant (n+1)^{2t}$. Hence the two norms are bounded by $(n+1)^{2t}A^n$ and $(n+1)^{2t}A^n p^t$ respectively. Due to the size of n and t in this case, we have $(n+1)^{2t} \approx L_Q(1/3, 4c_t/3c_p)$ and thus the coefficient $(n+1)^{2t}$ is negligible (when Q tends to infinity) compared to the $L_Q(2/3)$ contribution of A^n and p^t. As a consequence, we recover the two usual bounds so this allows us to continue with the same analysis of NFS as before. We conclude that the asymptotic complexity of the extended NFS when $l_p = 1/3$ is, also:

$$L_Q(1/3, (128/9)^{1/3}).$$

Yet the $p = L_Q(1/3, c_p)$ case remains particular because we have to consider both the extended version of the NFS algorithm and the Function Field Sieve algorithm (FFS) from [JL06]. It is not our main point here to develop the analysis of this boundary

case in the FFS part but we need to know which algorithm gives the best asymptotic complexity depending on the constant c_p. The Figure 2 addresses this need by showing how complexities vary with c_p in this case. The intersection between the FFS and the NFS approach is at $c_p = \kappa$ with $\kappa = (16/9)^{1/3}$. It is also indicated which algorithm has to be chosen in each case: the FFS algorithm when c_p is smaller than κ, and the NFS algorithm when c_p is higher. This figure is the juxtaposition of two analyses: the NFS analysis performed above and the FFS analysis from [JL06].

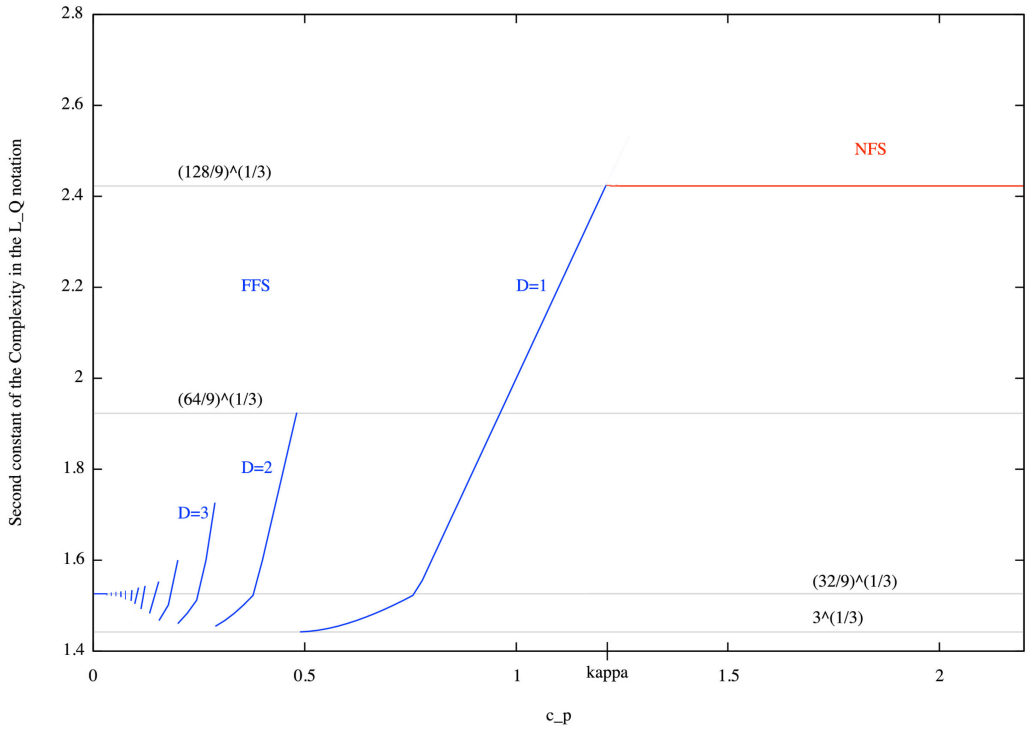

Figure 2: Asymptotic complexities at the $p = L_Q(1/3, c_p)$ boundary case. The graph represents the second constant c of the complexity $L_Q(1/3, c)$ as a function of c_p.

3.2 The $p = L_Q(l_p, c_p)$ Case with $2/3 \leqslant l_p \leqslant 1$

In this case, we modify both settings and sieving phase. First, the choice of the polynomials has to be changed, since the size of the coefficients of f_2 is too high when f_2 is defined as $f_1 + p$. In [JLSV06] the authors propose to select the polynomial f_2 using lattice reduction. We do not give details on this method since it does not affect our new algorithm developed further. Besides, we also consider another sieving space. Indeed, we can collect enough relations with a smaller sieving space than the one involved above. Setting $t = 2$ is sufficient, thus we only sieve on linear polynomials. The linear algebra and the individual discrete logarithms phases are left unchanged.

4 Applicable Discrete Logarithms for Pairing-Based Cryptography

Constructing pairing-based elliptic curves with a high security level implies taking into account the complexities of the various discrete logarithm algorithms that can be used. The traditional approach is to balance the complexity of a generic algorithm for computing discrete logarithms in the relevant subgroup of the elliptic curve and an estimate of the complexity of the NFS algorithm in the finite field where the pairing takes its values. This requires that $\sqrt{p} = L_Q(1/3, c)$, for some constant c. Equivalently, we have:

$$p = L_Q(1/3, 2c). \tag{2}$$

It is relevant to notice that this constraint imposed on curves give an indication on the form of the characteristic p, and that this explicit form permits conversely to estimate the actual complexity of computing discrete logarithms in the finite field considered. As a consequence of (2), we consider the case discussed in section 3.1.2. To determine which algorithm is applicable for computing discrete logarithms in $\mathbb{F}_Q$ with $p = L_Q(1/3, 2c)$, we have to evaluate the constant $2c$. As said in the introduction, we notice that current constructions select keys implicitly as if the complexity in the finite field was $L_Q(1/3, c)$ with a constant $c = (64/9)^{1/3}$. This leads to $2c \approx 3.845998854$, which is clearly higher than the boundary point κ. This points out that the extended NFS is the algorithm applicable here.

As a consequence, the actual choice of parameters and, in particular, the usual implicit assumption that $c = (64/9)^{1/3}$ are too pessimistic compared with the state-of-the-art. In fact, the analysis of section 3.1.2 shows that, using the best currently known variant of the Number Field Sieve algorithm, we have to choose $c = (128/9)^{1/3}$. We still have $2c > \kappa$.

5 SNFS Polynomials for Pairing-Based Finite Fields

5.1 Pairing-Based Finite Fields

Instead of proceeding as in Section 3 in the case of finite fields of general form we consider now the specificity of finite fields obtained with some particular curves. In practice, pairings require elliptic curves to be computationally very simple to use, and, often, not too difficult to generate. With this aim, families of such curves are frequently characterized by three simple polynomials, including P which defines after evaluation the characteristic of $\mathbb{F}_{p^n}$ where $\varphi : E \times E \to \mathbb{F}_{p^n}$ is the pairing considered, E a particular curve in the family and n its embedding degree. Several families have been proposed [FST10] and most of them have in common to set P as a polynomial of small degree and with constant coefficients. Until now, we consider the case of a particular family of curves where p the characteristic of $\mathbb{F}_{p^n}$ is given by the evaluation of such a polynomial. In other words, we consider a family where p can be written as:

$$p = P(u),$$

with P a polynomial of small degree λ and small coefficients and u small compared to p. We want to underline that λ is fixed beforehand and only depends on the family considered. Thus λ does not depend on p. In the following subsection 5.2, we explain how to use this sparse representation of p to lower the asymptotic heuristic complexity for the whole range of finite fields covered by the NFS – see the complete analysis in Section 6.

5.2 Choice of Polynomials for the SNFS Algorithm

We explain now how to use the specific structure of the polynomials characterizing pairing-friendly curves. Only a slight change has to be made in the algorithm described above: it concerns the choice of the two polynomials f_1 and f_2. We choose f_1 as an irreducible polynomial over $\mathbb{F}_p$, with degree equals to n, such as:

$$f_1(X) = X^n + R(X) - u$$

with $R(X)$ a polynomial of small degree and with coefficients 0, 1 or -1. Since we have $P(u) = p$, the size of the coefficients of f_1 is bounded by $p^{1/\lambda}$. Let us be more precise about the degree d_R of R. f_1 is a polynomial of degree n and has consequently approximatively one chance over n to be irreducible over the finite field: thus we need to keep enough degree of freedom concerning the choice of the coefficients of f_1. Hence we assume that the degree d_R is such as $d_R = O(\log n/\log 3)$. Since $3^{\log n/\log 3} = n$, this permits us to have enough choices for R, thus for f_1, and finally to obtain an irreducible polynomial [2].

Moreover, the second polynomial is chosen as follows:

$$f_2(X) = P(X^n + R(X)).$$

Indeed, f_2 has degree λn and the size of its coefficients is bounded by $O(\log(n)^\lambda)$. This comes from the $R(X)^\lambda$ term that appears in the decomposition of f_2, which provides the highest coefficients of f_2. Its coefficients are in fact bounded by $(d_R+1)^\lambda = O(\log(n)^\lambda)$. Furthermore, we have:

$$f_2(X) = P(f_1(X) + u) \equiv P(u) = p,$$

where $\equiv$ denotes equivalence $\bmod f_1(X)$.

Thus there exists a polynomial h such that $f_2(X) - p = h(X)f_1(X)$, and this implies that $f_2(X)$ is a multiple of $f_1(X)$ modulo p. Due to the fact that the $\gcd(f_1(X), f_2(X))$ is irreducible with degree n, they correctly define the commutative diagram previously drawn. The main interest of this choice is to keep small degrees while forcing a very small product of the two size of coefficients: the polynomials have respectively $(n, p^{1/\lambda})$ and $(\lambda n, O(\log(n)^\lambda))$ as degree and size of coefficients.

6 Asymptotic Heuristic Complexity

In order to analyze the asymptotic heuristic complexity of the algorithm described above, we first write the relations between n, p and $Q = p^n$ in the following form:

$$p = \exp\left(c_p(\log Q)^{l_p}(\log\log Q)^{1-l_p}\right), \qquad n = \frac{1}{c_p}\left(\frac{\log Q}{\log\log Q}\right)^{1-l_p}.$$

The parameters of the algorithm that appear in the analysis are the smoothness bound B, the degree of the elements in the sieving space $t-1$ and the bound A on the coefficients of these elements. We recall that we note λ the degree of the polynomial P mentioned in Section 5.

[2] Other possibilities are available for the polynomial R, all without any influence over the final asymptotic complexities. In the opposite way of our choice (small degree and constant coefficients), we could take R of constant degree d_R and with coefficients bounded by $O(n^{1/(dr+1)})$. An interesting configuration is to consider the intermediate case and to balance the degree of R with the size of its coefficients. If we force the coefficients to be bounded by d_r, the irreducibility of f_1 leads to the condition $d_r^{d_r} \approx n$ and finally to take $d_r = O(\log n/\log\log n)$. This impacts on the coefficients of f_2 which become bounded by $O((\log n/\log\log n)^\lambda)$ instead of $O(\log(n)^\lambda)$.

6.1 The $p = L_Q(l_p, c_p)$ Case with $1/3 \leqslant l_p < 2/3$

We assume that we can express t, A and B as

$$t = \frac{c_t}{c_p}\left(\frac{\log Q}{\log\log Q}\right)^{2/3-l_p}, \qquad A^t = L_Q(1/3, c_a c_t), \qquad B = L_Q(1/3, c_b)$$

where c_a, c_b and c_t will be determined later on.

In order to minimize the total runtime of the algorithm, we want to balance the complexities of the sieving phase and of the linear algebra phase. Since the total sieving space contains A^t elements, and the linear algebra phase costs approximately B^2 operations, we require that t, A and B satisfy $A^t = B^2$. This leads to the first condition:

$$c_b = \frac{c_a c_t}{2}. \tag{3}$$

Since we need to have enough good relations after sieving, we also require that $A^t\mathcal{P} \approx B$, where $\mathcal{P}$ denotes the probability that an element of the sieving space yields a good relation, *i.e.* the probability that its norms (in each of the two number fields) split into primes number smaller than B. Put together with the previous remark, this means:

$$B \approx 1/\mathcal{P}. \tag{4}$$

Let us note N_i the norm coming from the polynomial $h = a_{t-1}X^{t-1} + \cdots + a_0$ in $\mathbb{Q}[X]/(f_i(X))$ (for $i = 1$ and $i = 2$ to account for both sides). We can bound the two norms as follows. Keeping the notations of Section 3.1.2, N_1 is smaller than $\Theta A^n p^{t/\lambda}$, because f_1 is of degree n and its coefficients are bounded by $p^{1/\lambda}$ and h is of degree $t-1$ and its coefficients are bounded by A. Similarly we have N_2 smaller than $\Theta A^{\lambda n}\log(n)^{\lambda t}$. Thus, $\mathcal{P}$ is the probability that $\Theta^2\log(n)^{\lambda t}A^{(\lambda+1)n}p^{t/\lambda}$ splits into primes lower than B. Besides, the calculus of $A^{(\lambda+1)n}p^{t/\lambda}$ gives $L_Q(2/3, (\lambda+1)c_a + c_t/\lambda)$. We remark that both Θ^2 and $\log(n)^{\lambda t}$ are negligible in this case: in fact, both terms are smaller than $(n+1)^{4\lambda t}$ and $(n+1)^{4\lambda t} \approx L_Q(2/3 - l_p)$. We now make the usual heuristic hypothesis, and assume that $\mathcal{P}$ follows the theorem of Canfield, Erdös and Pomerance: a random number below $L_q(r,\gamma)$ splits into primes lower than $L_q(s,\delta)$ with probability $L_q(r-s, -\gamma(r-s)/\delta)$. As a result, after plugging in our values, we find that:

$$\mathcal{P} = L_Q\left(\frac{1}{3}, \frac{-1}{3c_b}((\lambda+1)c_a + c_t/\lambda)\right). \tag{5}$$

Putting together (4) and (5), we finally obtain the second condition involving the various constants:

$$3\,c_b^2 = (\lambda+1)c_a + c_t/\lambda. \tag{6}$$

We now want to minimize c_b under the two conditions (3) and (6). Hence, the complexity will be $L_Q(1/3, 2\,c_b^{min})$, where c_b^{min} is naturally the minimum we are looking for. Let us introduce two new variables μ and x and rewrite (3):

$$c_t = x, \qquad c_a = \mu x, \qquad c_b = \frac{\mu x^2}{2}.$$

Then (6) becomes:

$$3\left(\frac{\mu x^2}{2}\right)^2 = (\lambda+1)\mu x + \frac{x}{\lambda} \qquad \Leftrightarrow \qquad x^3 = \frac{4}{3}\cdot\left(\frac{(\lambda+1)\mu + 1/\lambda}{\mu}\right)$$

Minimizing $2\,c_b = \mu x^2$ is clearly equivalent to minimizing $(\mu x^2)^3$, so we calculate:

$$(\mu x^2)^3 = \left(\frac{4}{3}\right)^2 \cdot \left(\frac{(\lambda+1)^2\mu^2 + 1/\lambda^2 + 2\mu(\lambda+1)/\lambda}{\mu}\right).$$

Finally, forcing the derivate of the right member with respect to μ to vanish implies $(\lambda+1)^2\mu^2 - 1/\lambda^2 = 0$ and at the end $\mu = 1/(\lambda(\lambda+1))$. As a result, $(2\,c_b^{min})^3 = (\mu x^2)^3 = (64/9)\cdot(\lambda+1)/\lambda$. Thus, the complexity of the algorithm in this case is:

$$L_Q\left(\frac{1}{3}, \left(\frac{64}{9}\cdot\frac{\lambda+1}{\lambda}\right)^{1/3}\right).$$

As soon as $\lambda \geqslant 2$, the complexity is clearly better than the one in the general case, which is $L_Q(1/3, (128/9)^{1/3})$.

6.2 The $p = L_Q(2/3, c_p)$ Case

In this case, we consider a family of algorithms indexed by the degree $t-1$ of the polynomials we are sieving on and we compute the asymptotic complexity of each algorithms. Figure 3 shows which algorithm has to be chosen, depending on the constant c_p, in order to get the best asymptotic complexity. The analysis made here follows exactly the previous one, except that the round-off error in t is no longer negligible. This explains why the final complexity varies with c_p. We continue the analysis in the general case while the two extreme cases $t \to \infty$ and $t = 2$ are discussed further. When t tends to infinity, we recover the asymptotic complexity of the $p = L_Q(l_p, c_p)$ case with $1/3 \leqslant l_p < 2/3$. Furthermore, the asymptotic complexity is minimal for the choice of p that are compatible with $t = 2$, *i.e.* that allows sieving on linear polynomials. Thus we explicitly compute the complexity in this case.

Sieving on Polynomials of Degree $t-1$

We assume that we can express A and B as:

$$A = L_Q(1/3, c_a) \qquad \text{and} \qquad B = L_Q(1/3, c_b).$$

The sieving space contains in this case A^t elements since the polynomials involved are of degree $t-1$. Thus, balancing the size of the sieving space and the runtime of the linear algebra we deduce $c_a = 2c_b/t$. Keeping the same notations as above and neglecting again the Θ and $\log(n)^\lambda$ terms, we can write the two norms $N_1 = A^n p^{(t-1)/\lambda}$ and $N_2 = A^{\lambda n}$. So the product of the two norms is $A^{(\lambda+1)n} p^{(t-1)/\lambda}$, which can also be written as:

$$N_1 \cdot N_2 = L_Q\left(\frac{2}{3}, \frac{2(\lambda+1)c_b}{c_p t} + \frac{(t-1)c_p}{\lambda}\right).$$

In order to get enough relations we force B to be equal to the inverse of the probability of smoothness $\mathcal{P}$, *i.e.* $B = L_Q\left(\frac{1}{3}, \frac{1}{3c_b}\left(\frac{2(\lambda+1)c_b}{c_p t} + \frac{(t-1)c_p}{\lambda}\right)\right)$. This leads to the following equation:

$$3\,c_b^2 = \frac{2(\lambda+1)c_b}{c_p t} + \frac{(t-1)c_p}{\lambda}.$$

Consequently, the sieving on polynomials of degree $t-1$ has complexity $L_Q(1/3, C_\lambda(c_p))$ with:

$$C_\lambda(c_p) = 2\,c_b = \frac{2}{3}\left(\frac{\lambda+1}{c_p t} + \sqrt{\left(\frac{\lambda+1}{c_p t}\right)^2 + \frac{3(t-1)c_p}{\lambda}}\right). \tag{7}$$

This has to be compared with the asymptotic complexity in the General Number Field Sieve (GNFS) for the same case which is $L_Q(1/3, C(c_p))$ with:

$$C(c_p) = \frac{2}{3}\left(\frac{2}{c_p t} + \sqrt{\left(\frac{2}{c_p t}\right)^2 + 3(t-1)c_p}\right).$$

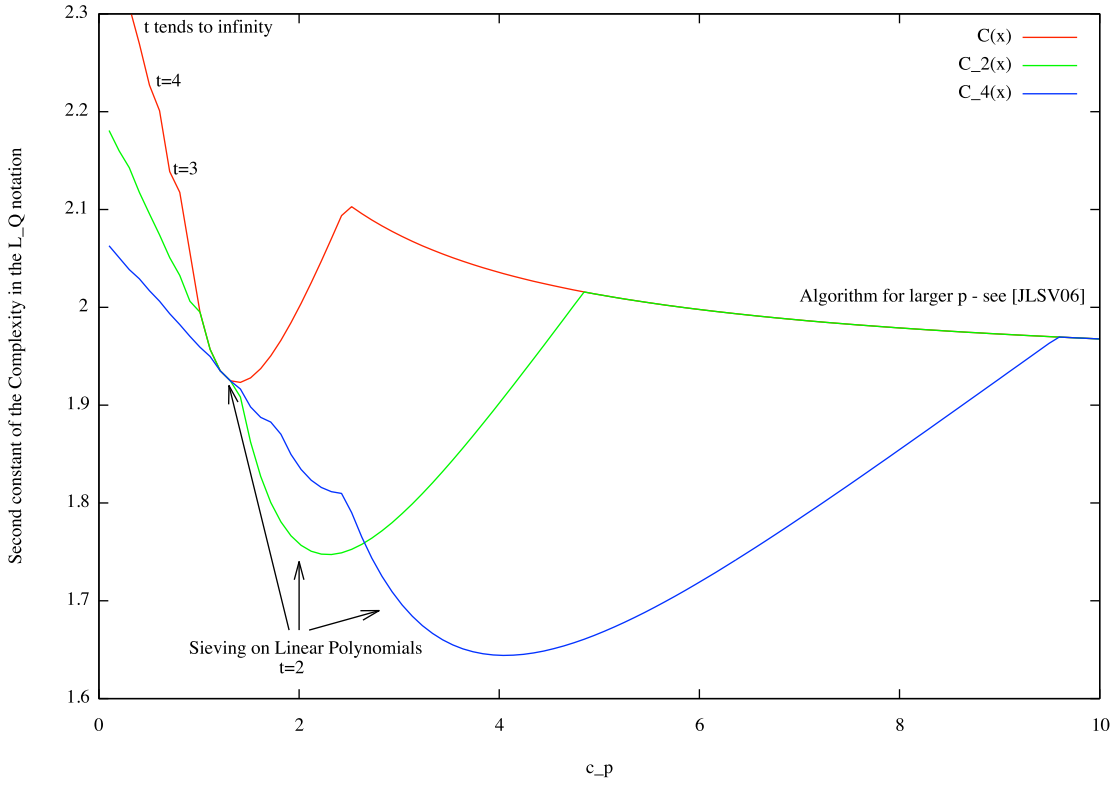

Figure 3: Asymptotic complexities $L_Q(1/3, C(c_p))$, $L_Q(1/3, C_2(c_p))$ and $L_Q(1/3, C_4(c_p))$ as a function of c_p with $p = L_Q(2/3, c_p)$. The red curve shows the variation of the second constant of the complexity for the GNFS while the green and blue ones present the amelioration obtained by our SNFS in two cases $\lambda = 2$ and $\lambda = 4$. The degree $t-1$ of the elements in the sieving space is also indicated for the red curve.

In Figure 3 we have plotted the constant $C(c_p)$ which determines the complexity $L_Q(1/3, C(c_p))$ as a function of the constant c_p. The red curve represents the constant $C(c_p)$ obtained with the GNFS [JLSV06] while the other ones are obtained with our SNFS for $\lambda = 2$ (green curve) and $\lambda = 4$ (blue curve). Those values of λ correspond respectively to the family of MNT curves and to the family of Barreto-Naehrig elliptic curves.

Splicing the $p = L_Q(2/3, c_p)$ Case to the $p = L_Q(l_p, c_p)$ Case with $l_p < 2/3$

We consider c_p as a variable and compute the value of c_p which minimize the complexity $L_Q(1/3, C_\lambda(c_p))$ given in the previous subsection 6.2. We note it again c_p^{min}. $C_\lambda(c_p)$ comes

to a minimum when:

$$\frac{\lambda+1}{c_p^2 t} = \frac{-2(\lambda+1)^2 t^{-2} c_p^{-3} + 3(t-1)\lambda^{-1}}{2\sqrt{\left(\frac{\lambda+1}{c_p t}\right)^2 + \frac{3(t-1)c_p}{\lambda}}}$$

$$\begin{aligned} &\Leftrightarrow \quad 4(\lambda+1)^4\lambda^2 + 12t^2(t-1)\lambda(\lambda+1)^2 c_p^3 = (3(t-1)t^2 c_p^3 - 2\lambda(\lambda+1))^2 \\ &\Leftrightarrow \quad t^2(t-1)c_p^3 = (8/3)\lambda(\lambda+1)^2. \end{aligned}$$

Thus we take:

$$c_p^{min} = \left(\frac{8}{3} \cdot \frac{\lambda(\lambda+1)^2}{(t-1)t^2}\right)^{1/3}.$$

As a consequence, putting together with (7) we obtain:

$$c_b = 2\left(\frac{1}{3^2} \cdot \frac{t-1}{t} \cdot \frac{\lambda+1}{\lambda}\right)^{1/3}.$$

We conclude that the minimal complexity of the sieving on polynomials of degree $t-1$ in this case is:

$$L_Q\left(\frac{1}{3}, \left(\frac{64}{9} \cdot \frac{t-1}{t} \cdot \frac{\lambda+1}{\lambda}\right)^{1/3}\right). \tag{8}$$

If $p = L_Q(2/3, c_p)$ can only be written with a constant c_p close to zero, it is better in practice to write it as $p = L_Q(l_p, c'_p)$ with $l_p < 2/3$ and a constant c'_p higher than c_p, and to apply afterwards the previous algorithm. Nonetheless, if we fix $p = L_Q(2/3, c_p)$, when c_p tends to zero the best choice is to force t to tend to infinity (Figure 3). Theoretically, it is interesting to see that $t \to \infty$ yields the expected limit:

$$L_Q\left(\frac{1}{3}, \left(\frac{64}{9} \cdot \frac{\lambda+1}{\lambda}\right)^{1/3}\right)$$

which is the asymptotic complexity of the $p = L_Q(l_p, c_p)$ case with $1/3 \leqslant l_p < 2/3$.

Sieving on Linear Polynomials

Let us go back to the minimal complexity that appears in (8). Considering that the asymptotic complexity in GNFS [JLSV06] for the same case is $L_Q(1/3, (128/9).(t-1/t)^{1/3})$, we remark that for each algorithm our variant multiplied by a factor $\frac{\lambda+1}{2\lambda}$ the cube of the second constant of the complexity in the L_q notation. In particular, this gives an interesting result when we are sieving on linear polynomials, *i.e.* when $t-1=1$. Replacing t by 2 in (8), we find that sieving on linear polynomials leads to the following complexity:

$$L_Q\left(\frac{1}{3}, \left(\frac{32}{9} \cdot \frac{\lambda+1}{\lambda}\right)^{1/3}\right).$$

This has to be compared with the asymptotic complexity in GNFS [JLSV06] for the same case which is $L_Q(1/3, (64/9)^{1/3})$. Again, as soon as $\lambda \geqslant 2$, the complexity of our SNFS is clearly better than the one in the general case.

6.3 Algorithm for Larger p

The $p = L_Q(l_p, c_p)$ Case with $2/3 < l_p < 1$

We recall that sieving on linear polynomials in this case is sufficient. Let A be the bound on the coefficients of the polynomials we are sieving over, and B the smoothness bound. Again, balancing the size of the sieving space A^2 and the runtime of the linear algebra B^2 leads to $A = B$. We assume that we can express B as $B = L_Q(1/3, c_b)$. The product of the two norms is as usual bounded by $\Theta^2 \log(n)^\lambda B^{(\lambda+1)n} p^{1/\lambda}$. Due to the size of n in this case, Θ^2 and $\log(n)^\lambda$ are negligible compared to $B^{(\lambda+1)n} p^{1/\lambda}$. Let us develop the logarithm of this new bound: $\log(B^{(\lambda+1)n} p^{1/\lambda}) = n \log B + \lambda n \log B + (1/\lambda) \log p$. First, we remark that we have $n \log B = \log L_Q(4/3 - l_p) = \log L_Q(l)$ with $l < 2/3$. Setting besides:

$$\lambda n = c_l \left(\frac{\log Q}{\log \log Q} \right)^{1/3} \tag{9}$$

with c_l to be determined later on, we obtain $\lambda n \log B + (1/\lambda) \log p = \log(L_Q(2/3, c_b c_l + 1/c_l))$. Thus B^n is negligible compared to $B^{\lambda n} p^{1/\lambda}$. Moreover, $B^{\lambda n} p^{1/\lambda}$ comes to a minimum when:

$$c_l = 1/\sqrt{c_b}. \tag{10}$$

The product of the two norms is so bounded by $L_Q(2/3, 2\sqrt{c_b})$. Hence the probability that it is B-smooth is $L_Q(1/3, -2/(3\sqrt{c_b}))$. As usual, we equalize B with the inverse of the probability. This yields $c_b = 2\sqrt{c_b}$ and then $c_b = (4/9)^{1/3}$. Putting this value in (10), we get $c_l = (2/3)^{1/3}$. The constraint (9) over λ becomes:

$$\lambda = \frac{1}{n} \left(\frac{2 \log Q}{3 \log \log Q} \right)^{1/3}.$$

Finally, the asymptotic complexity of the algorithm for this degree λ is:

$$L_Q(1/3, (32/9)^{1/3}).$$

This has to be compared with the complexity of GNFS in the same case, which is $L_Q(1/3, (64/9)^{1/3})$.

The Boundary Case : $l_p = 2/3$

When $l_p = 2/3$ matters are more complicated since B^n is no more negligible. The product of the norms is now rewritten as $L_Q(2/3, c_b(c_l + 1/c_p) + 1/c_l)$. Again, this is minimized at $c_l = 1/\sqrt{c_b}$. However, it now becomes $L_Q(2/3, 2\sqrt{c_b} + c_b/c_p)$ and the probability of smoothness is $L_Q(1/3, -(1/3).(2/\sqrt{c_b}) + 1/c_p))$. Equating the opposite of the second constant to c_b yields $(3c_b + 1/c_p)^2 = 4/c_b$ and finally:

$$9c_b^3 - \frac{6}{c_p} c_b^2 + \frac{1}{c_p^2} c_b - 4 = 0.$$

When c_p tends to infinity, we recover the $(32/9)^{1/3}$ constant in the complexity.

The $p = L_Q(1, c_p)$ Case

The analysis follows exactly the previous one, except that we have a simpler expression of the extension degree $n = 1/c_p$. Setting again $\lambda = c_p (2 \log Q)^{1/3} (3 \log \log Q)^{-1/3}$, we obtain the final asymptotic complexity $L_Q(1/3, (32/9)^{1/3})$. In particular, this applies on finite fields of prime order, since $n = 1$ implies that p can be written as $p = L_p(1, 1)$. We recall that the original SNFS applies on such fields of prime order and has the same complexity $L_Q(1/3, (32/9)^{1/3})$ – see [Sch06, Sch08].

7 Conclusion

In this paper, we adapted the Special Number Field Sieve to compute discrete logarithms in $\mathbb{F}_{p^n}$ when p is obtained by evaluation of a polynomial with small coefficients. More precisely, for $p = L_Q(l_p, c_p)$ with $1/3 \leqslant l_p < 2/3$ our variation yields a complexity of $L_Q(1/3, [(64/9) \cdot (\lambda+1)/\lambda)]^{1/3})$ where λ is the small degree of the polynomial P which gives the characteristic p after evaluation. This should be compared with the previous $L_Q(1/3, (128/9)^{1/3})$ in the General High Degree Number Field Sieve. Likewise, for $p = L_Q(2/3, c_p)$ we make the asymptotic heuristic complexity drop from $L_Q(1/3, (64/9)^{1/3})$ to $L_Q(1/3, [(32/9) \cdot (\lambda+1)/\lambda]^{1/3})$. For larger p ($p = L_Q(l_p, c_p)$ with $2/3 < l_p < 1$), it goes down from $L_Q(1/3, (64/9)^{1/3})$ to $L_Q(1/3, (32/9)^{1/3})$ for some λ with a suitable size compared with p.

References

[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[BLS03] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. On the selection of pairing-friendly groups. In *Selected Areas in Cryptography*, pages 17–25, 2003.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *J. Cryptology*, 17(4):297–319, 2004.

[BN05] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography*, pages 319–331, 2005.

[CC03] Jae Choon Cha and Jung Hee Cheon. An identity-based signature from gap Diffie-Hellman groups. In *Public Key Cryptography*, pages 18–30, 2003.

[FST10] David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. *J. Cryptology*, 23(2):224–280, 2010.

[Gor93] Daniel M. Gordon. Discrete logarithms in *gf(p)* using the number field sieve. *SIAM J. Discrete Math.*, 6(1):124–138, 1993.

[JL06] Antoine Joux and Reynald Lercier. The function field sieve in the medium prime case. In *EUROCRYPT*, pages 254–270, 2006.

[JLSV06] Antoine Joux, Reynald Lercier, Nigel P. Smart, and Frederik Vercauteren. The number field sieve in the medium prime case. In *CRYPTO*, pages 326–344, 2006.

[Jou04] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. *J. Cryptology*, 17(4):263–276, 2004.

[Kal97] Michael Kalkbrener. An upper bound on the number of monomials in determinants of sparse matrices with symbolic entries. In *Mathematica Pannonica 8*, pages 73–82, 1997.

[KM05] Neal Koblitz and Alfred Menezes. Pairing-based cryptography at high security levels. In *IMA Int. Conf.*, pages 13–36, 2005.

[LV01] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *J. Cryptology*, 14(4):255–293, 2001.

[Nat03] National Institute of Standards and Technology. Special publication 800-56: Recommendation on key establishment schemes, Draft 2.0, 2003.

[Pat02] Kenneth G. Paterson. Id-based signatures from pairings on elliptic curves. *IACR Cryptology ePrint Archive*, 2002:4, 2002.

[Sch06] Oliver Schirokauer. The number field sieve for integers of low weight. *IACR Cryptology ePrint Archive*, 2006:107, 2006.

[Sch08] Oliver Schirokauer. The impact of the number field sieve on the discrete logarithm problem in finite fields. *Algorithmic Number Theory*, 44, 2008.

[SK03] Ryuichi Sakai and Masao Kasahara. Id based cryptosystems with pairing on elliptic curve. *IACR Cryptology ePrint Archive*, 2003:54, 2003.