The Number Field Sieve in the Medium Prime Case

Antoine Joux[1,3], Reynald Lercier[1,2], Nigel Smart[4], and Frederik Vercauteren[5⋆]

[1] DGA
[2] CELAR
Route de Laillé, 35170 Bruz, France
Reynald.Lercier@m4x.org
[3] Université de Versailles St-Quentin-en-Yvelines
PRISM
45, avenue des Etats-Unis, 78035 Versailles Cedex, France
Antoine.Joux@m4x.org
[4] Dept. Computer Science, University of Bristol
MVB, Woodland Road, Bristol, BS8 1UB, United Kingdom
nigel@cs.bris.ac.uk
[5] Department of Electrical Engineering, University of Leuven
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
frederik.vercauteren@esat.kuleuven.be

Abstract. In this paper, we study several variations of the number field sieve to compute discrete logarithms in finite fields of the form $\mathbb{F}_{p^n}$, with p a medium to large prime. We show that when n is not too large, this yields a $L_{p^n}(1/3)$ algorithm with efficiency similar to that of the regular number field sieve over prime fields. This approach complements the recent results of Joux and Lercier on the function field sieve. Combining both results, we deduce that computing discrete logarithms have heuristic complexity $L_{p^n}(1/3)$ in all finite fields. To illustrate the efficiency of our algorithm, we computed discrete logarithms in a 120-digit finite field $\mathbb{F}_{p^3}$.

1 Introduction

Today's public key cryptosystems usually rely on either integer factorisation or discrete logarithms in finite fields or on elliptic curves. In this paper, we consider discrete logarithm computations in finite fields of the form $\mathbb{F}_{p^n}$, with $n > 1$ and p a medium to large prime. For a long time, these fields have been mostly ignored in cryptosystems, and the security of their discrete logarithm problems are much less known than the security in prime fields $\mathbb{F}_p$ or fields of fixed characteristic such as $\mathbb{F}_{2^n}$ or $\mathbb{F}_{3^n}$. Indeed, in $\mathbb{F}_p$ it is well-known that the best available algorithm is the number field sieve, first introduced in [7] and further studied in [18, 20, 21, 12, 19]. Similarly, in fixed characteristic and n tending to infinity, the best available algorithm is the function field sieve, first introduced in [3] and further studied in [4, 11, 8]. Both algorithms have complexity $L_{p^n}(1/3)$.

⋆ Postdoctoral Fellow of the Research Foundation - Flanders (FWO - Vlaanderen)

However, until very recently for the intermediate case $\mathbb{F}_{p^n}$, no general $L_{p^n}(1/3)$ algorithm was known and the best available approach had complexity $L_{p^n}(1/2)$ (see [2, 1]). With the advent of pairing-based and torus-based cryptography, the security of discrete logarithms in $\mathbb{F}_{p^n}$ is a growing concern. Recently, two very different methods have been proposed to deal with some of these fields. The first approach is based on rational torus representation and was proposed by Granger and Vercauteren [9]. It was effectively used by Lercier and Vercauteren in [15]. The second approach, by Joux and Lercier [13], proposes a family of algorithms which are based on the function field sieve and yields complexity $L_{p^n}(1/3)$ where applicable.

In this paper, we introduce several variations of the number field sieve algorithm that depend on the relative size of p and n. The main difference with existing algorithms lies in the construction of the number fields and the choice of sieving space: for p large compared to $L_{p^n}(2/3)$, we use a new polynomial selection algorithm and for p small compared to $L_{p^n}(2/3)$, we sieve over elements of degree higher than one. Furthermore, we show that these variations can fill the gap which was left open by [13]. As a consequence, we conclude that discrete logarithm computation has heuristic complexity $L_{p^n}(1/3)$ for all finite fields, which greatly improves the $L_{p^n}(1/2)$ complexity of [2, 1].

The paper is organised as follows: in Section 2 we describe our setup and the basic number field sieve variation that we are considering. In Section 3 we show how the basic algorithm can be modified in order to cover all the finite fields left out by [13]. In Section 4 we describe the mathematical background required to make this description rigorous and in Section 5 we give a precise heuristic analysis of the algorithm. Finally in Section 6 we report on experiments obtained with an actual implementation of this algorithm.

2 Our Basic Variation on the Number Field Sieve

The number and function field sieve algorithms are both index calculus algorithms that search for multiplicative identities using smooth objects over well-chosen smoothness bases. In the number field sieve, the smoothness bases contain ideals of small norm. Since we are using ideals, some important mathematical technicalities arise when transforming these multiplicative identities into linear equations involving logarithms. For the sake of simplicity, we defer these technicalities to Section 4. When dealing with index calculus algorithms, the complexities are usually expressed using the following notation:

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

where log denotes natural logarithm. When the constant c is not given explicitly, the notation $L_q(\alpha)$ is often used. In particular, for the prime field $\mathbb{F}_p$ and for extension fields $\mathbb{F}_{p^n}$ with fixed characteristic p, the number field sieve and the function field sieve respectively yield $L_p(1/3, (64/9)^{1/3})$ and $L_{p^n}(1/3, (32/9)^{1/3})$ algorithms. Moreover, in [13] it was shown that variations of the function field sieve yield $L_{p^n}(1/3)$ algorithms for $\mathbb{F}_{p^n}$ as long as $p \leq L_{p^n}(1/3)$. In this paper, we consider the complementary case: $p \geq L_{p^n}(1/3)$.

2.1 Setup, Sieving and Linear Algebra

Our basic variation addresses finite fields $\mathbb{F}_{p^n}$ with $p = L_{p^n}(2/3, c)$ and c near $2 \cdot (1/3)^{1/3}$. Recall that the regular number field sieve algorithm over $\mathbb{F}_p$ starts by choosing two polynomials $f_1, f_2 \in \mathbb{Z}[X]$ with a common root in $\mathbb{F}_p$. In our basic variation, we generalise this to $\mathbb{F}_{p^n}$ as follows: first, f_1 is chosen as a degree n polynomial, with very small coefficients and irreducible over $\mathbb{F}_p$. Then, we set f_2 equal to the polynomial $f_1 + p$. Since $f_2 \equiv f_1 \bmod p$, both polynomials clearly have a common root in $\mathbb{F}_{p^n}$ (in fact all of them are equal).

Let $K_1 \simeq \mathbb{Q}[X]/(f_1(X)) \cong \mathbb{Q}[\theta_1]$ and $K_2 \cong \mathbb{Q}[X]/(f_2(X)) \cong \mathbb{Q}[\theta_2]$ be the two number fields defined respectively by f_1 and f_2, i.e. θ_1 and θ_2 are roots of f_1 and f_2 in $\mathbb{C}$. Choose a smoothness bound B and a sieve limit S and consider all pairs (a, b) of coprime integers, with $|a| \leq S$ and $|b| \leq S$, such that $a - b\theta_1$ and $a - b\theta_2$ both have B-smooth norms. After some post-processing described in Section 4, which involves adding unit contributions or Schirokauer maps, each such pair yields a linear equation between "logarithms of ideals" in the smoothness bases. Since the number of small norm ideals involved on each side is smaller than $n\pi(B)$, with $\pi(x)$ the number of primes smaller than x, it suffices to collect $2n\pi(B)$ equations during the sieving phase. Once the sieving phase is complete, we solve the resulting sparse system of linear equations modulo the cardinality of $\mathbb{F}_{p^n}^*$, or more precisely, modulo a large factor of this cardinality and recover logarithms of ideals of small norm. There are two options for this large factor: either we take it prime, but then we need to factor $p^n - 1$ (which would not increase the total complexity) or we simply take it composite without small factors, since all operations we perform modulo a large prime factor remain valid for such a composite factor. Finally, small prime factors of $p^n - 1$ can be dealt with separately using a combination of the Pohlig-Hellman and Pollard rho algorithm.

2.2 Individual Discrete Logarithms

Once the two steps described above, sieving and linear algebra, have been performed, we obtain the logarithms of the ideals of the smoothness bases. In order to complete the computation, an additional step is required, namely computing the logarithm of random elements in the finite field. We propose a classical approach based on "special-$\mathfrak{q}$" descent, which is similar to the approach in [12] for the case of logarithms over a prime field.

Represent $\mathbb{F}_{p^n}$ as $\mathbb{F}_p[t]/(f_1(t))$ and assume we want to compute the discrete logarithm of some element $y \in \mathbb{F}_{p^n}$. First, we search for an element of the form $z = y^i t^j$ for some $i, j \in \mathbb{N}$ with the following additional properties:

1. after lifting z to the number field K_1 (also denoted by z), its norm factors into primes smaller than some bound $B_1 \in L_{p^n}(2/3, 1/3^{1/3})$,
2. the norm of the lift of z should be squarefree, implying that only degree one prime ideals will appear in the factorisation of (z).

Assuming that z does not belong to a strict subfield[1] of $\mathbb{F}_{p^n}$, nothing prevents such a squarefree factorisation to occur. Therefore, we make the usual heuristic hypothesis and assume that the norm has a squarefree factorisation into small primes with probability similar to that of a random number of the same size. According to [10], this probability can be expressed as a product of two terms. The first term is the usual probability of smoothness, without the squarefree condition. The second term, comes from the squarefree condition and quickly tends to $6/\pi^2$. Since the constant $6/\pi^2$ vanishes into the $o(1)$ of the L notation, we simply ignore this technicality in the remainder of the paper.

Whilst the first condition on the element z is standard, the second condition is required to guarantee that only degree one prime ideals will appear in the factorisation of (z). This condition is necessary since during the sieving phase, we only computed logarithms of degree one prime ideals. Since squared factors in the norm could correspond to higher degree ideals, we would not know the corresponding logarithms.

After finding an adequate candidate, we factor the principal ideal generated by z into degree one prime ideals of small norm. Note that there will be several prime ideals not contained in the factor base, since their norm is allowed to be bigger than the smoothness bound B. To compute the logarithm of such an ideal $\mathfrak{q}$, we start a special-$\mathfrak{q}$ descent as follows: we sieve on pairs (a, b), where a and b are chosen to ensure that $\mathfrak{q}$ divides $a - b\theta_1$ in the number field K_1. After finding a pair (a, b) such that the norm of $a - b\theta_1$ (divided by the norm of $\mathfrak{q}$) and the norm of $a - b\theta_2$ factor into primes smaller than $B_2 < B_1$, we iterate the descent lowering the bound at each step until it becomes lower than B. At this point, all the discrete logarithms are known and we can backtrack to recover the logarithm of each of the initial $\mathfrak{q}$ and consequently the logarithm of z. Note that during the descent, we will need to consider special-$\mathfrak{q}$ ideals in both number fields.

2.3 Practical Improvements

Galois extensions If possible, we should choose f_1 such that K_1 is Galois over $\mathbb{Q}$. Let $\mathrm{Gal}(K_1/\mathbb{Q})$ be its Galois group, then since p is inert in K_1, we obtain an isomorphism $\mathrm{Gal}(K_1/\mathbb{Q}) \simeq \mathrm{Gal}(\mathbb{F}_q/\mathbb{F}_p)$ (see [17, Chapter I, 9.6]) which implies that K_1 has to be a cyclic number field of degree n. The first major improvement for Galois K_1, is that the factor basis associated with K_1 can be n times smaller. We refer to Section 4.3 for a detailed description. The second improvement is related to finding an adequate z in the individual logarithm phase. Assuming that n is prime, the prime ideals lying above a rational prime l are of two types only: either (l) is inert or splits into degree one prime ideals. In the former case, we simply have to compute logarithms in $\mathbb{F}_p$, which is a much smaller finite field, so we can neglect its cost. In the latter case, we proceed as before. For Galois K_1 of prime extension degree n, we therefore only need to find a z with sufficiently smooth norm and can ignore the squarefree condition.

[1] Of course, when z belongs to a strict subfield, we are in an easier case of the algorithm, since it then suffices to compute the logarithm in $\mathbb{F}_p^*$ of z w.r.t. the norm of a generator of $\mathbb{F}_{p^n}$.

Choice of polynomials Instead of simply setting $f_2 = f_1 + p$, we can in fact take any polynomial f_2 such that $f_1 | f_2$ over the finite field $\mathbb{F}_p$. In particular, if f_1 defines a cyclic number field, f_2 should be chosen to maximise the automorphism group of K_2. For instance, for all primes $p \equiv 2, 5 \bmod 9$, $f_1 = x^6 + x^3 + 1$ is irreducible over $\mathbb{F}_p$ and by choosing $f_2 = x^6 + (p+1)x^3 + 1$, K_2 will have a non-trivial automorphism group of order 2. Constructing a K_2 with large automorphism group for general p remains an open problem.

Since f_1 normally has tiny coefficients, the coefficients of the polynomial f_2 will be much larger than those of f_1. To balance the size of the norms we compute during the algorithm, there are basically two approaches: unbalance the size of a and b of the elements we sieve over or change 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, c, is of order $\sqrt{p}$. Write $c \equiv c_1/c_2 \bmod p$ with c_1 and c_2 also of order $\sqrt{p}$ and define $f_2 \equiv c_2 f_1 \bmod p$. The polynomial f_2 is no longer monic, but its coefficients are $O(\sqrt{p})$ instead of $O(p)$.

Individual logarithms Instead of directly decomposing z as a product of ideals, it is useful in practice to start by writing z as a fraction of the form:

$$\frac{\sum a_i t^i}{\sum b_i t^i},$$

where the coefficients a_i and b_i are of the order of $\sqrt{p}$.

More precisely, this is done by reducing the following lattice L, where the generating vectors are in columns and where the algebraic numbers in the first line stand for the subcolumns of their coordinates:

$$L = \begin{pmatrix} \mathbf{z} & \mathbf{tz} & \mathbf{t^2z} & \cdots & \mathbf{t^{n-1}z} & \mathbf{p} & \mathbf{pt} & \mathbf{pt^2} & \cdots & \mathbf{pt^{n-1}} \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & 0 \end{pmatrix}.$$

Clearly, any vector in this lattice represents a fraction equal to z, where the numerator is encoded by the upper half of the vector and the denominator by the lower half. Since the determinant of the lattice is p^n and the dimension $2n$, we expect that lattice reduction can find a short vector of norm $\sqrt{p}$. Note that this is only a practical expectation in small dimension that does not hold when n becomes too large.

3 Extension to Other Fields

The analysis (cf. Section 5) of the previous algorithm yields a $L_{p^n}(1/3)$ complexity when p has the right size compared to p^n. In this section, we discuss the possible adaptations to larger and smaller values of p.

3.1 Smaller Values of p

When p is smaller than in the basic case, we encounter a major problem, when sieving over the (a, b) pairs. Indeed, we would like to keep the norms of $a - b\theta_1$ and $a - b\theta_2$ below $L_{p^n}(2/3)$, however, since p is small, n is larger than before and there are less than $L_{p^n}(1/3)$ relations. To summarise, the possible sieving space is too small and we cannot collect enough relations. Thus, we need to enlarge the sieving space. A simple technique is to sieve over elements of higher degree. Instead of pairs (a, b), we consider $(t + 1)$-tuples $(a_0, \cdots, a_t)$ and compute the norms of $\sum_{i=0}^{t} a_i\theta_1^i$ and $\sum_{i=0}^{t} a_i\theta_2^i$. Each of these norms can be obtained through a resultant computation of two polynomials $A(x) = \sum_{i=0}^{t} a_i x^i$ and $f_1(x)$ (resp. $f_2(x)$). It is well-known that the resultant can be obtained as the determinant of an $(n + t) \times (n + t)$ matrix formed of t columns containing the coefficients of f_1 and n columns containing the coefficients of A. Thus, we can clearly bound the norm by $(n + t)^{n+t} {B_a}^n {B_f}^t$, where B_a is an upper bound on the absolute values of the a_i and B_f a similar bound on the coefficients of f_1 (resp. f_2).

3.2 Larger Values of p

When p is larger than in the basic case, the basic polynomial construction is no longer usable, because due to the addition of p, a coefficient of f_2 is larger than $L_{p^n}(2/3)$. In order to lower the size of the coefficients, we need to change the polynomial construction and to use polynomials with higher total degree. This is not completely surprising, since the number field sieve in the prime field case uses polynomials with total degree increasing with p. The most difficult part is to find a polynomial construction which does not somehow fail in the individual logarithm phase. We propose the following simple construction. Start from a polynomial f_0 of degree n with small coefficients (and optionally cyclic Galois group when possible). Then, we choose a constant W and let $f_1(x) = f_0(x+W)$, the largest coefficient of f_1 is around W^n. Then, using lattice reduction, we search for a polynomial f_2 of degree D and coefficients smaller than W^n, such that $f_1 | f_2 \bmod p$. This can be done by reducing the lattice L, where the generating vectors are in columns

$$L = \left(\mathbf{f_1(x)\ xf_1(x)\ x^2f_1(x)} \cdots \mathbf{x^{D-n}f_1(x)\ p\ px\ px^2} \cdots \mathbf{px^D} \right)$$

It is clear that every linear combination of the columns results in a polynomial which is divisible by f_1 modulo p. Furthermore, the dimension of the lattice L is $D + 1$ and its determinant is p^{D+1}. We know that for lattices of dimension s and determinant $\mathcal{D}$, LLL [14] outputs a vector shorter than $2^{s/4}\mathcal{D}^{1/s}$. We want this vector to encode a polynomial different from f_1. This can be ensured if the vector is smaller than the encoding of f_1. Thus, we need:

$$2^{(D+1)/4} p^{n/(D+1)} \le W^n.$$

Since the term $2^{(D+1)/4}$ can be hidden in the $o(1)$ of the L representation, the optimal size of the coefficients in f_1 and f_2 is $R = W^n \approx p^{n/(D+1)}$.

4 Mathematical Aspects

In this section, we describe the mathematical background required to make our algorithm rigorous. As before, let $\mathbb{F}_q$ be a finite field with $q = p^n$ elements, with p prime. Given $h = g^x \in \mathbb{F}_q^*$ and a large prime divisor l of $q-1$, we show how to recover $x \bmod l$.

We will work in several number fields of the form $K = \mathbb{Q}[\theta]$ for some algebraic integer θ with minimal polynomial $f(X) \in \mathbb{Z}[X]$. In the basic case, all these number fields will be of degree n over $\mathbb{Q}$, such that p remains prime in the ring of integers $\mathcal{O}_K$ of K, i.e. $\mathcal{O}_K/(p) \cong \mathbb{F}_q$. In general, we also need number fields of degree $m > n$, such that there exists a prime ideal $\mathfrak{Q}$ of degree n lying above p, i.e. $\mathcal{O}_K/\mathfrak{Q} \cong \mathbb{F}_q$. To construct such a number field, it simply suffices for $f(X) \bmod p$ to have an irreducible factor of degree n. Denote by $\varphi_{\mathfrak{Q}}$ the surjective ring homomorphism obtained by dividing out the ideal $\mathfrak{Q}$, then for any element $x \in \mathcal{O}_K$ we define $\overline{x} = \varphi_{\mathfrak{Q}}(x)$.

4.1 Factoring Ideals in $\mathcal{O}_K$

Recall that in the algorithm we look for coprime pairs of integers (a, b) such that the principal ideal $(a - b\theta)$ factors into prime ideals of small norm, i.e. of norm smaller than some predefined bound B. The possible prime ideals occurring in the factorisation of $(a - b\theta)$ are given by the following lemma [6, Lemma 10.5.1].

Lemma 1. *Let $K = \mathbb{Q}[\theta]$ and (a, b) coprime integers, then any prime ideal $\mathfrak{p}$ which divides $a - b\theta$ either divides the index $f_\theta = [\mathcal{O}_K : \mathbb{Z}[\theta]]$ or is of degree one.*

Therefore, let $\mathcal{F}$ consist of degree one prime ideals of norm smaller than B and the finitely many prime ideals that divide the index f_θ, then we try to decompose $(a - b\theta)$ over $\mathcal{F}$. Each degree one prime ideal is generated by $(p_i, \theta - c_{p_i})$ with p_i a rational prime smaller than B and c_{p_i} a root of $f(X) \bmod p_i$.

Furthermore, finding the decomposition of the ideal $(a-b\theta)$ into prime ideals is equally easy. First compute its norm $\mathrm{N}_{K/\mathbb{Q}}(a - b\theta) = b^{\deg(f)} f(a/b)$ and test whether $\mathrm{N}_{K/\mathbb{Q}}(a - b\theta)$ is B-smooth. If it is, write $\mathrm{N}_{K/\mathbb{Q}}(a - b\theta) = \prod_i p_i^{e_i}$; now we need to make the following distinction:

- For rational primes $p_i \nmid f_\theta$, we obtain that $\mathfrak{p}_i = (p_i, \theta - c_{p_i})$ with $c_{p_i} \equiv a/b \bmod p_i$ will occur in the ideal factorisation of $(a - b\theta)$ and that the $\mathfrak{p}_i$-adic valuation is precisely e_i.
- For rational primes $p_i | f_\theta$, use Algorithm 4.8.17 in Cohen [6].

At the end, we therefore have several pairs of coprime integers (a, b) with corresponding ideal decompositions

$$(a - b\theta) = \prod_i \mathfrak{p}_i^{e_i} \; . \tag{1}$$

For p smaller than in the basic case, we will need to sieve over $(t+1)$-tuples of coprime integers $(a_0, a_1, \ldots, a_t)$ such that the principal ideal $(\sum_{i=0}^{t} a_i \theta^i)$ factors into prime ideals of norm smaller than B. The following lemma is an easy generalisation of Lemma 1.

Lemma 2. *Let $K = \mathbb{Q}[\theta]$ and $(a_0, \ldots, a_t)$ a $(t+1)$-tuple of coprime integers, then any prime ideal $\mathfrak{p}$ that divides $(\sum_{i=0}^{t} a_i\theta^i)$ either divides the index $f_\theta = [\mathcal{O}_K : \mathbb{Z}[\theta]]$ or is of degree $\leq t$.*

To find the decomposition of the ideal $(\sum_{i=0}^{t} a_i\theta^i)$, we compute its norm as $\mathrm{N}_{K/\mathbb{Q}}(\sum_{i=0}^{t} a_i\theta^i) = \mathrm{Res}(\sum_{i=0}^{t} a_i X^i, f(X))$, with Res the resultant. For each prime factor p_i not dividing the index f_θ, we proceed as follows: each irreducible factor of $\gcd(\sum_{i=0}^{t} a_i X^i, f(X))$ over $\mathbb{F}_{p_i}$, corresponds to an ideal $\mathfrak{p}_i$ lying above p occurring in the ideal decomposition. Furthermore, if the gcd itself is irreducible, the $\mathfrak{p}_i$-adic valuation is simply $m/\deg \mathfrak{p}_i$; if not, we use Algorithm 4.8.17 in Cohen [6]. For prime factors dividing the index, we proceed as described above.

4.2 From Ideals to Elements

We now show how to transform the relations involving ideals into multiplicative relations involving elements only. After mapping these relations to $\mathbb{F}_q^*$ using $\varphi_{\mathfrak{Q}}$ and taking logarithms, we obtain linear equations between logarithms of elements in $\mathbb{F}_q^*$ modulo the prime factor l.

We make a distinction between two cases: the simple case where K has class number one and computable unit group, the general case.

***K* with Class Number One and Computable Unit Group** Since the class number equals one, every ideal I in $\mathcal{O}_K$ is principal, i.e. there exists $\gamma_I \in \mathcal{O}_K$ with $I = (\gamma_I)$. The ideal decomposition $(a - b\theta) = \prod_i \mathfrak{p}_i^{e_i}$ then leads to

$$a - b\theta = u \prod_i \gamma_i^{e_i}$$

with $(\gamma_i) = \mathfrak{p}_i$ and u a unit in $\mathcal{O}_K$. Let (r_1, r_2) be the signature of K, then the unit group $\mathcal{O}_K^* \cong \mu(K) \times \mathbb{Z}^{r_1+r_2-1}$, with $\mu(K) = \langle u_0 \rangle$ a finite cyclic group of order v. Assuming we can compute a system of fundamental units $u_1, \ldots, u_r$ with $r = r_1 + r_2 - 1$, we can write $u = u_0^{n_0} u_1^{n_1} \cdots u_r^{n_r}$. In this setting we thus obtain r logarithmic maps λ_i for $i = 1, \ldots, r$ defined by

$$\lambda_i : \mathcal{O}_K^* \to \mathbb{Z} : u \mapsto n_i \,,$$

and a logarithmic map $\lambda_0 : \mathcal{O}_K^* \mapsto \mathbb{Z}/v\mathbb{Z} : u \mapsto n_0$. Finally, we obtain the decomposition

$$a - b\theta = \prod_{i=0}^{r} u_i^{\lambda_i(u)} \prod_i \gamma_i^{e_i} \,.$$

Applying $\varphi_{\mathfrak{Q}}$ and taking logarithms of both sides then leads to

$$\log_g(a - b\overline{\theta}) \equiv \sum_{i=0}^{r} \lambda_i(u) \log_g \overline{u}_i + \sum_i e_i \log_g \overline{\gamma}_i \bmod q - 1 \,. \quad (2)$$

General *K* Here we assume that the large prime factor l of $q-1$ does not divide the class number $h(K)$, which constitutes a very minor restriction. To obtain a relation between elements, we raise both sides of the ideal decomposition $(a-b\theta)=\prod_i \mathfrak{p}_i^{e_i}$ to the power $h=h(K)$ and obtain

$$(a-b\theta)^h = u\prod_i \delta_i^{e_i} \tag{3}$$

with u a unit and $\delta_i \in \mathcal{O}_K$ with $(\delta_i)=\mathfrak{p}_i^h$. Note that the left hand side denotes the h-th power of the element (not the ideal) $a-b\theta$. Furthermore, we will never compute the elements δ_i, but only need that these exist and correspond to the ideals $\mathfrak{p}_i$.

It is tempting to apply $\varphi_{\mathfrak{Q}}$ to both sides and take logarithms as in the easy case, leading to

$$h\log_g(a-b\overline{\theta}) \equiv \log_g \overline{u} + \sum_i e_i \log_g \overline{\delta}_i \bmod q-1\,.$$

The problem with this approach is that for each equation, the unit u will be different, so $\log_g \overline{u}$ is a new unknown for every equation, and thus we will never obtain a linear system of full rank. Note that in the easy case, we could express u as a product of fundamental units, which effectively circumvented this problem.

To solve this problem we follow Schirokauer [18]: since it is sufficient to compute the discrete logarithm modulo a large prime $l|q-1$, we need not work with the whole unit group $\mathcal{O}_K^*$, but only modulo l-th powers of units, i.e. $\mathcal{O}_K^*/(\mathcal{O}_K^*)^l$. Clearly, for $u\in(\mathcal{O}_K^*)^l$, we have $\log_g \overline{u}\equiv 0 \bmod l$. Instead of defining logarithmic maps on the whole unit group, Schirokauer defines such maps on $\mathcal{O}_K^*/(\mathcal{O}_K^*)^l$.

For simplicity we will assume that l does not ramify in K. Consider the set $\Gamma_K=\{\gamma\in\mathcal{O}_K \mid \mathrm{N}_{K/\mathbb{Q}}(\gamma)\not\equiv 0 \bmod l\}$, and note that the elements $a-b\theta$ are in Γ, since they are smooth and thus $l\nmid \mathrm{N}_{K/\mathbb{Q}}(a-b\theta)$. Also note that Γ_K is multiplicative and contains the unit group $\mathcal{O}_K^*$.

Let $\varepsilon=l^D-1$, with D the least common multiples of the irreducible factors of $f(X) \bmod l$, then for all $\gamma\in\Gamma_K$ we have

$$\gamma^\varepsilon \equiv 1 \bmod l\,.$$

Define the map λ from Γ_K to $l\mathcal{O}_K/l^2\mathcal{O}_K$ by $\lambda(\gamma)=(\gamma^\varepsilon-1)+l^2\mathcal{O}_K$. Fix a basis $\{lb_i+l^2\mathcal{O}_K : i=1,\ldots,n\}$ for $l\mathcal{O}_K/l^2\mathcal{O}_K$ with $\{lb_i+l^2\mathcal{O}_K : i=1,\ldots,r\}$ a basis for $\lambda(\mathcal{O}_K^*)$ and define maps $\lambda_i:\Gamma_K\to\mathbb{Z}/l\mathbb{Z}$ by

$$(\gamma^\varepsilon-1)\equiv l\sum_{i=1}^n \lambda_i(\gamma)b_i \bmod l^2\,.$$

Note that $\lambda(\gamma\cdot\gamma')=\lambda(\gamma)+\lambda(\gamma')$ and similarly $\lambda_i(\gamma\cdot\gamma')=\lambda_i(\gamma)+\lambda_i(\gamma')$, so these maps are logarithmic on Γ_K. As a consequence, the maps λ_i for $i=1,\ldots,n$ are in fact homomorphisms from $\mathcal{O}_K^*$ to $\mathbb{Z}/l\mathbb{Z}$. Consider the homomorphism

$$\overline{\lambda}:\mathcal{O}_K^*/(\mathcal{O}_K^*)^l \to (\mathbb{Z}/l\mathbb{Z})^r : u\mapsto(\lambda_1(u),\ldots,\lambda_r(u))\,.$$

We have the following trivial lemma which is a special case of Schirokauer's theorem.

Lemma 3. *Assuming that $\overline{\lambda}$ is injective, $u \in \mathcal{O}_K^*$ is an l-th power if and only if $\overline{\lambda}(u) = 0$.*

If we furthermore assume that $\mathcal{O}_K^*$ does not contain the primitive l-th roots of unity, then $\mathcal{O}_K^*/(\mathcal{O}_K^*)^l$ has l^r elements and thus $\overline{\lambda}$ defines an isomorphism. Therefore, there exist units $u_1, \ldots, u_r \in \mathcal{O}_K^*$ such that $\lambda_i(u_i) = 1$ and $\lambda_i(u_j) = 0$ for $i \neq j$. Note that these units are not unique, since they are only determined modulo $(\mathcal{O}_K^*)^l$. Any unit $u \in \mathcal{O}_K$ can thus be written as

$$u = \prod_{i=1}^{r} u_i^{\lambda_i(u)} \cdot \xi^l \tag{4}$$

for some unit ξ.

In Equation (3) we now modify the elements δ_i by multiplying with a well defined unit

$$\delta_i' = \delta_i \cdot \prod_{j=1}^{r} u_j^{-\lambda_j(\delta_i)} ,$$

such that $\lambda_j(\delta_i') = 0$ for $j = 1, \ldots, r$. Note that δ_i' still generates $\mathfrak{p}_i^h$.

Rewriting Equation (3) then leads to $(a - b\theta)^h = u' \prod_i (\delta_i')^{e_i}$, with u' a unit. Since we have constructed δ_i' such that $\lambda_j(\delta_i') = 0$ for $j = 1, \ldots, r$, we have $h\lambda_j(a - b\theta) = \lambda_j(u')$. Rewriting the unit u' as in (4) finally leads to the equation

$$(a - b\theta)^h = \xi_{a,b}^l \cdot \prod_{i=1}^{r} u_i^{h\lambda_i(a-b\theta)} \cdot \prod_i (\delta_i')^{e_i} ,$$

for some $\xi_{a,b} \in \mathcal{O}_K^*$. Applying $\varphi_{\mathfrak{Q}}$ and taking logarithms of both sides modulo l then gives

$$\log_g(a - b\overline{\theta}) \equiv \sum_{i=0}^{r} \lambda_i(a - b\theta) \log_g \overline{u}_i + \sum_i e_i h^{-1} \log_g \overline{\delta}_i' \bmod l . \tag{5}$$

Note that $h^{-1} \log_g \overline{\delta}_i'$ and $\log_g \overline{u}_i$ correspond precisely to the virtual logarithms of ideals and Schirokauer maps introduced in [12] and [19], but are now given as logarithms of the reduction of specific elements in $\mathfrak{p}_i$ and $\mathcal{O}_K^*$. Furthermore, note that these values are well defined modulo l.

4.3 Exploiting Automorphisms

In this section we show that if the number field K has a non-trivial automorphism group $\mathrm{Aut}(K)$, the size of the factor basis $\mathcal{F}$ can be reduced by a factor of $\#\mathrm{Aut}(K)$.

Let K be a number field of degree n with non-trivial automorphism group $\mathrm{Aut}(K)$ and assume p is inert in K. Clearly, the prime ideal (p) is invariant under each $\phi \in \mathrm{Aut}(K)$. Therefore, each automorphism ϕ defines an automorphism $\overline{\phi}$ of $\mathbb{F}_q$ by reduction modulo p. Furthermore, since p is unramified, this map will be injective, so $\overline{\mathrm{Aut}}(K) \subset \mathrm{Gal}(\mathbb{F}_q/\mathbb{F}_p)$, implying that $\mathrm{Aut}(K)$ has to be cyclic and $\#\mathrm{Aut}(K)|n$.

Let $A = \#\mathrm{Aut}(K)$ and $d = n/A$ and denote by $\psi \in \mathrm{Aut}(K)$ the unique automorphism such that $\psi(x) \equiv x^{p^d} \bmod p$ for all $x \in \mathcal{O}_K$. Then $\mathrm{Aut}(K) = \langle \psi \rangle$ and define $\psi_k = \psi^k$ for $k = 0, \ldots, A-1$.

Write $\mathcal{F} = \mathcal{F}_e \cup \mathcal{F}_u$ where $\mathcal{F}_u$ contains all unramified degree one prime ideals and $\mathcal{F}_e$ the others. Note that $\mathcal{F}_e$ is tiny, since these prime ideals have to divide the discriminant of f. Since $\psi_k(\mathfrak{p})$ is a prime ideal of the same degree and ramification index as $\mathfrak{p}$, we can partition $\mathcal{F}_u$ into A disjunct sets $\mathcal{F}_{u,k}$ for $k = 0, \ldots, A-1$ such that $\mathfrak{p} \in \mathcal{F}_{u,0} \Leftrightarrow \psi_k(\mathfrak{p}) \in \mathcal{F}_{u,k}$. The decomposition (1) can then be rewritten as

$$(a - b\theta) = \prod_{\mathfrak{p}_i \in \mathcal{F}_e} \mathfrak{p}_i^{e_i} \prod_{k=0}^{A-1} \prod_{\mathfrak{p}_i \in \mathcal{F}_{u,0}} \psi_k(\mathfrak{p}_i)^{e_{i,k}} \,,$$

where most of the e_i and $e_{i,k}$ will be zero.

Let δ_i be a generator of the principal ideal $\mathfrak{p}_i^h$, then clearly $\psi_k(\delta_i)$ is a generator of $(\psi_k(\mathfrak{p}_i))^h$ and $\log_g \overline{\psi_k(\delta_i)} \equiv p^{dk} \log_g \overline{\delta}_i \bmod q-1$. Rewrite Equation (3) as

$$(a - b\theta)^h = u' \prod_{i \in [\mathcal{F}_e]} \delta_i^{e_i} \prod_{k=0}^{A-1} \prod_{i \in [\mathcal{F}_{u,0}]} \psi_k(\delta_i)^{e_{i,k}} \,,$$

for some unit u' and with $[\cdot]$ denoting the index set of a set. Note that we no longer have the equality $\lambda_j(u') = h\lambda_j(a - b\theta)$, since $\lambda_j(\psi_k(\delta_i))$ will not be zero. However, we can explicitly compute

$$\lambda_j(u') \equiv h\lambda_j(a - b\theta) - \sum_{i \in [\mathcal{F}_e]} e_i \lambda_j(\delta_i) - \sum_{k=0}^{A-1} \sum_{i \in [\mathcal{F}_{u,0}]} e_{i,k} \lambda_j(\psi_k(\delta_i)) \bmod l \,.$$

The equivalent of Equation (5) then becomes

$$\begin{aligned} \log_g(a - b\overline{\theta}) \equiv & \sum_{j=0}^{r} h^{-1} \lambda_j(u') \log_g \overline{u}_j + \sum_{i \in [\mathcal{F}_e]} h^{-1} e_i \log_g \overline{\delta}_i \\ & + \sum_{i \in [\mathcal{F}_{u,0}]} \left(h^{-1} \sum_{k=0}^{A-1} p^{dk} e_{i,k} \right) \log_g \overline{\delta}_i \bmod l \,. \end{aligned} \tag{6}$$

Note that the number of unknowns in the right hand side of the above equation now only amounts to $\#\mathcal{F}_{u,0} + \#\mathcal{F}_e + r$, which is roughly $\#\mathrm{Aut}(K)$ times smaller than in the previous section. However, this approach is only feasible when we are able to compute the class number h. Furthermore, for each ideal $\mathfrak{p}_i \in \mathcal{F}_{u,0}$ we now have to explicitly compute a δ_i that generates $\mathfrak{p}_i^h$. and all the Schirokauer maps $\lambda_j(\psi_k(\delta_i))$ for $j = 1, \ldots, r$, $k = 0, \ldots, A-1$ and all i. Clearly none of these issues arise when K has class number one and computable unit group.

5 Asymptotic Heuristic Complexity

In this section we summarise the heuristic complexity of the different variations described in Sections 2 and 3. The details of this complexity analysis are given in Appendix A. Note that the basic algorithm is a special case of the algorithm given in Section 3.1, so we do not treat it separately.

The precise complexity depends on the ratio of p compared to p^n and we have the following cases:

- p can be written as $L_q(l_p, c)$ with $1/3 < l_p < 2/3$, then the algorithm given in 3.1 has complexity
$$L_q(1/3, (128/9)^{1/3}) \,.$$
- p can be written as $L_q(2/3, c)$ for a constant c, then the algorithm given in 3.1 has complexity
$$L_q(1/3, 2c') \quad \text{with} \quad c' = \frac{4}{3}\left(\frac{3t}{4(t+1)}\right)^{1/3} \,,$$
where we sieve over $(t+1)$-tuples where the degree t is taken as the natural number closest to the real root of $3c^3t(t+1)^2 - 32 = 0$.
- p can be written as $L_q(2/3, c)$ for a constant c, then the algorithm given in 3.2 has complexity
$$L_q(1/3, 2c') \quad \text{with} \quad 9c'^3 - \frac{6}{c}c'^2 + \frac{1}{c^2}c' - 8 = 0 \,.$$
- p can be written as $L_q(l_p, c)$ with $l_p > 2/3$, then the algorithm given in 3.2 has complexity
$$L_q(1/3, (64/9)^{1/3}) \,.$$

In Figure 1 we have plotted the constant c' which determines the complexity $L_q(1/3, 2c')$ as a function of the constant c. The plot also shows for which c to switch from algorithm 3.1 to algorithm 3.2.

6 Numerical Experiments

We modified Joux and Lercier's discrete logarithm C-implementation in order to handle a 120-digit realistic experiment. More precisely, we consider a finite field of the form $\mathbb{F}_{p^3}$ with

$$p = \lfloor 10^{39}\pi \rfloor + 2622 = 3141592653589793238462643383279502886819 \,,$$

the smallest 40-digit integer than $10^{39}\pi$ such that $(p-1)/2$ is prime. Its multiplicative group has $p^3 - 1$ elements, which factors as

$$2 \times 13 \times 19 \times 199 \times 4177 \times 212145751 \times 547465820443 \times 337091202666800863 \times l \times (p-1)/2 \,,$$

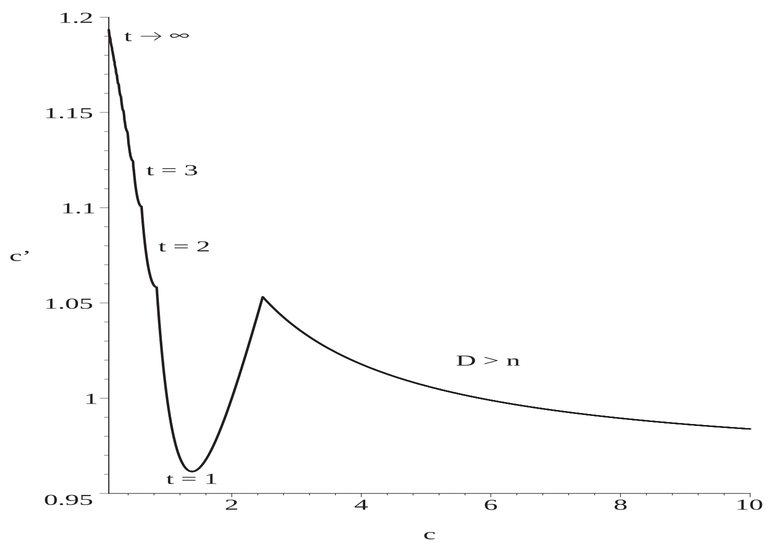

Fig. 1. Asymptotic complexity $L_q(1/3, 2c')$ (note factor 2) as a function of c with $p \in L_q(1/3, c)$. Also indicated are the degree t of the elements in the sieving space and the degree $D \geq n$ of the number field K_2.

where $l = 1227853277188311599267416539617839$. The hardest part is of course to compute discrete logarithms modulo the 110-bit factor l since smaller factors can be easily handled with the Pollard-rho algorithm and discrete logarithms modulo $p-1$ are the same as discrete logarithms of norms in $\mathbb{F}_p$. To solve this problem, let $\mathbb{Q}[\theta_1]$ and $\mathbb{Q}[\theta_2]$ be the two number fields defined respectively by

$$f_1(X) = X^3 + X^2 - 2X - 1 \quad \text{and} \quad f_2(X) = f_1(X) + p\,,$$

where we have $\mathbb{F}_{p^3} \simeq \mathbb{F}_p[t]/(f_1(t))$. With these settings, $\mathbb{Q}[\theta_1]$ is a cubic cyclic number field, the Galois group of which is given by $\mathrm{Aut}(\mathbb{Q}[\theta_1]) = \{\theta_1 \mapsto \theta_1, \theta_1 \mapsto \theta_1^2 - 2, \theta_1 \mapsto -\theta_1^2 - \theta_1 + 1\}$, and a system of fundamental units by $u_1 = \theta_1 + 1$ and $u_2 = \theta_1^2 + \theta_1 - 1$. $\mathbb{Q}[\theta_2]$ has signature $(1,1)$, thus we will have to apply one single Schirokauer logarithmic map λ_1 to our relations to deal with obstructions in the $\mathbb{Q}[\theta_2]$ side.

We construct smoothness bases with 1 000 000 prime ideals as follows,

- in the $\mathbb{Q}[\theta_1]$ side, we include 899 999 prime ideals, but only 300 000 are meaningful due to the Galois action,
- in the $\mathbb{Q}[\theta_2]$ side, we include 700 000 prime ideals.

The lattice sieving considers only algebraic integers $a + b\theta_2$ which are already divisible by a prime ideal in $\mathbb{Q}[\theta_2]$, such that norms to be smoothed in $\mathbb{Q}[\theta_2]$ are 150 bit integers (once removed the known prime ideal) and norms in $\mathbb{Q}[\theta_1]$ are 110 bit integers (an important tip to improve the running time with such polynomials is to unbalance the size of a versus the size of b). The sieving took 12 days on a 1.15 GHz 16-processors HP AlphaServer GS1280.

We then had to compute the kernel of a $1\,163\,482 \times 793\,188$ matrix, the coefficients of which are mostly equal modulo ℓ to ± 1, $\pm p$ or $\pm p^2$. We thus first modified the Joux-Lercier structured gaussian elimination step to reduce such a system to $450\,246$ equations in $445\,097$ unknowns with $44\,544\,016$ non null entries. Time needed for this on one processor was only a few minutes. Then, the critical phase was the final computation via Lanczos's algorithm. Our parallelised version of this algorithm (modified for such matrices too) took one additional week. At the end, since $\mathbb{Q}[\theta_1]$ is principal, we may check that we have discrete logarithms of generators of ideals in the smoothness bases, for instance,

$$\begin{aligned}
(t^2+t+1)^{(p^3-1)/l} &= G^{294066886450155961127467122432171}, \\
(t-3)^{(p^3-1)/l} &= G^{364224563635095380733340123490719}, \\
(3\,t-1)^{(p^3-1)/l} &= G^{468876587747396380675723502928257},
\end{aligned}$$

where $G = g^{(p^3-1)/11592682025741777397154621558414 84\,l}$ and $g = -2t+1$.

In the final step, we took as a challenge $\gamma = \sum_{i=0}^{2} (\lfloor \pi \times p^{i+1} \rfloor \bmod p) t^i$. We first easily computed its discrete logarithm in basis g modulo $(p^3-1)/l$,

$$38895389158901518975845922936941184677534991099612214604576972713861472869102 82477328.$$

To obtain a complete result, we expressed

$$\gamma = \frac{-9098798035595952934 7\,t^2 - 114443008248522156910\,t + 154493664373341271998}{94912764441570771406\,t^2 - 120055569809711861965\,t - 81959619964446352567},$$

where numerator and denominator are both smooth once considered as algebraic integers in $\mathbb{Q}[\theta_1]$. Using a three level tree with 80 special-$\mathfrak{q}$ ideals, we recovered the discrete logarithm modulo l, namely 110781190155780903592153105706975. Each special-$\mathfrak{q}$ sieving took 10 minutes or a total of 14 hours.

7 Conclusion

In this paper, we have presented a new variation of the number field sieve algorithm to compute discrete logarithms in $\mathbb{F}_{p^n}^*$. For $p > L_{p^n}(1/3)$, our variation yields a complexity of $L_{p^n}(1/3)$. For smaller values of p, the function field sieve algorithm described in [13] also gives $L_{p^n}(1/3)$ complexity. As a consequence, we have $L_{p^n}(1/3)$ heuristic algorithms to compute discrete logarithms in all finite fields $\mathbb{F}_{p^n}$. This should be compared to the previous $L_{p^n}(1/2)$ algorithms given in [1, 2]. Another major advantage is that this algorithm has a fast post-processing phase for individual logarithms, once the main algorithm terminates. This is extremely useful for applications which require the computation of many logarithms in the same field. To give an example, this was required by the identity based cryptosystem of Maurer and Yacobi [16].

Acknowledgement

The authors would like to thank Henri Cohen for providing extensive references on the existence and construction of cyclic number fields.

References

1. L. Adleman and J. DeMarrais. A subexponential algorithm for discrete logarithms over all finite fields. In D. Stinson, editor, *Proceedings of CRYPTO'93*, volume 773 of *Lecture Notes in Comput. Sci.*, pages 147–158. Springer, 1993.
2. L. Adleman and J. DeMarrais. A subexponential algorithm for discrete logarithms over all finite fields. *Math. Comp.*, 61(203):1–15, 2003.
3. L.M. Adleman. The function field sieve. In *Algorithmic Number Theory, Proceedings of the ANTS-I conference*, volume 877 of *Lecture Notes in Comput. Sci.*, pages 108–121, 1994.
4. L.M. Adleman and M.A. Huang. Function field sieve method for discrete logarithms over finite fields. In *Information and Computation*, volume 151, pages 5–16. Academic Press, 1999.
5. E. R. Canfield, Paul Erdős, and Carl Pomerance. On a problem of Oppenheim concerning "factorisatio numerorum". *J. Number Theory*, 17(1):1–28, 1983.
6. H. Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1993.
7. D.M. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM J. Discrete Math.*, 6(1):124–138, 1993.
8. R. Granger, A. Holt, D. Page, N. Smart, and F. Vercauteren. Function field sieve in characteristic three. In D. Buell, editor, *Algorithmic Number Theory, Proceedings of the ANTS-VI conference*, volume 3076 of *Lecture Notes in Comput. Sci.*, pages 223–234. Springer, 2004.
9. R. Granger and F. Vercauteren. On the discrete logarithm problem on algebraic tori. In V. Shoup, editor, *Proceedings of CRYPTO'2005*, volume 3621 of *Lecture Notes in Comput. Sci.*, pages 66–85. Springer, 2005.
10. A. Ivić and G. Tenenbaum. Local densities over integers free of large prime factors. *Quart. J. Math. Oxford Ser. (2)*, 37(148):401–417, 1986.
11. A. Joux and R. Lercier. The function field sieve is quite special. In C. Fieker and D. Kohel, editors, *Algorithmic Number Theory, Proceedings of the ANTS-V conference*, volume 2369 of *Lecture Notes in Comput. Sci.*, pages 431–445. Springer, 2002.
12. A. Joux and R. Lercier. Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the gaussian integer method. *Math. Comp.*, 72:953–967, 2003.
13. A. Joux and R. Lercier. The function field sieve in the medium prime case. In S. Vaudenay, editor, *Proceedings of EUROCRYPT'2006*, volume 4004 of *Lecture Notes in Comput. Sci.*, pages 254–270. Springer, 2006.
14. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982.
15. R. Lercier and F. Vercauteren. Discrete logarithms in $\mathbb{F}_{p^{18}}$ - 101 digits. NMBRTHRY mailing list, June 2005.
16. U.M. Maurer and Y. Yacobi. A non-interactive public-key distribution system. *Des. Codes Cryptogr.*, 9(3):305–316, 1996.
17. J. Neukirch. *Algebraic number theory*, volume 322 of *Grundlehren der Mathematischen Wissenschaften*. Springer-Verlag, Berlin, 1999.
18. O. Schirokauer. Discrete logarithms and local units. *Philos. Trans. Roy. Soc. London Ser. A*, 345(1676):409–423, 1993.
19. O. Schirokauer. Virtual logarithms. *J. Algorithms*, 57(2):140–147, 2005.

20. O. Schirokauer, D. Weber, and T. Denny. Discrete logarithms: the effectiveness of the index calculus method. In *Algorithmic Number Theory, Proceedings of the ANTS-II conference*, volume 1122 of *LNCS*, pages 337–361. Springer, Berlin, 1996.
21. D. Weber. Computing discrete logarithms with the general number field sieve. In *Algorithmic Number Theory, Proceedings of the ANTS-II conference*, volume 1122 of *LNCS*, pages 391–403. Springer, Berlin, 1996.

A Asymptotic Heuristic Complexity

A.1 Basic Algorithm

In order to analyse the heuristic asymptotic complexity of the basic algorithm described in Section 2 we first assume that we can write the following relations between $q = p^n$, p and n, for some constant c to be determined later on:

$$n = \frac{1}{c} \cdot \left(\frac{\log q}{\log \log q}\right)^{1/3}, \qquad p = \exp\left(c \cdot \sqrt[3]{\log^2 q \cdot \log \log q}\right).$$

We further assume that the sieve limit S and the smoothness bound B are chosen equal and expressed as:

$$S = B = \exp\left(c' \cdot \sqrt[3]{\log q \cdot \log^2 \log q}\right),$$

for some constant c'. Since f_1 has small coefficients, say bounded by b_0, the norm of $a - b\theta_1$ is smaller than $nb_0B^n = B^{n+o(1)}$. Similarly, the norm of $a - b\theta_2$ is smaller than $p \cdot B^{n+o(1)}$. Thus, the product of these two norms is smaller than $p \cdot B^{2n+o(1)} = L_q(2/3, c + 2c'/c)$. We make the usual heuristic hypothesis and assume that the probability for each value of this product to split into primes lower than B follows the well-known theorem of Canfield, Erdös and Pomerance [5]: a random number below $L_q(r, \gamma)$ is $L_q(s, \delta)$-smooth with probability $L_q(r - s, -\gamma(r - s)/\delta)$. As explained above, in the squarefree factorisation case, this probability is lower, by a factor $6/\pi^2$ which we can neglect in the L-notation. Plugging in our values, we find that the product of the norms is smooth with probability $L_q(1/3, -(1/3) \cdot (c/c' + 2/c))$. In order to minimise the total runtime, we want to roughly balance the complexities of the sieving phase and of the linear algebra phase. This is achieved when $c' = (1/3) \cdot (c/c' + 2/c)$ and thus:

$$c' = \frac{1}{3}\left(\frac{1}{c} + \sqrt{3c + c^{-2}}\right).$$

The heuristic complexity is $L_q(1/3, 2c')$ and it varies depending on c. It is minimal for $c = 2 \cdot (1/3)^{1/3}$, where $c' = 2 \cdot (1/3)^{2/3}$. At this minimal point, the complexity is $L_q(1/3, (64/9)^{1/3})$, thus identical to the complexity of the number field sieve over prime fields.

A.2 Algorithm for Smaller p

When considering our algorithmic variants for smaller values of p, two cases are possible. Either p is written $L_q(2/3, c)$ as before or it is of the form $L_q(l_p, c)$ with $1/3 < l_p < 2/3$. In the first case, the overall complexity is $L_q(1/3)$ with a constant which depends on c, in the second case, the complexity is $L_q(1/3, (128/9)^{1/3})$. The parameters of the algorithms are the smoothness bound B, the degree t of the elements we are sieving over and the bound on the coefficients of these elements S. Clearly, the norm on the f_1 side is bounded by $S^{n+o(1)}$ and the norm on the f_2 side is bounded by $S^{n+o(1)}p^{t+o(1)}$. Thus the product of the norms is bounded by $S^{2n+o(1)}p^{t+o(1)}$. Choose for t the nearest integer to:

$$\frac{c_t}{c} \cdot \left(\frac{\log q}{\log\log q}\right)^{2/3-l_p},$$

for a constant c_t to be determined later on. Choose the sieve bound as:

$$S = \exp\left(c_S c \cdot \log^{l_p-1/3} q \cdot \log^{4/3-l_p} \log q\right).$$

Then, the total sieving space contains S^t elements, where

$$S^t = \exp\left((c_S c_t + o(1)) \cdot \log^{1/3} q \cdot \log^{2/3} \log q\right)$$

when $l_p < 2/3$. For the $l_p = 2/3$ case, the round-off error in t is no longer negligible, which explains why the complexity varies with c. We continue the analysis for $l_p < 2/3$, the $l_p = 2/3$ case is discussed in the next subsection and summarised in Figure 1. Let the smoothness bound be $B = L_q(1/3, c')$ as before. Rewrite the bound on the product of norms as $L_q(2/3, 2c_S + c_t)$, then the smoothness probability is

$$Pr = L_q(1/3, -(1/3) \cdot (2c_S/c' + c_t/c')).$$

In order to choose c_S and c_t, we need to equate the runtime of the sieving, i.e. the size of the sieving space S^t, with the time of the linear algebra, i.e. B^2, and also with B/Pr. This implies $S^{t/2} = B = 1/Pr$. Translating this into equations involving the various constants, we find:

$$c' = \frac{1}{3}\left(\frac{2c_S}{c'} + \frac{c_t}{c'}\right) = \frac{c_S c_t}{2}.$$

As a consequence, we deduce:

$$c' = \sqrt{\frac{2c_S + c_t}{3}} = \frac{c_S c_t}{2}.$$

We now write $c_t = x$, $c_S = \mu x$ and minimise c'. With this notation:

$$\frac{(2\mu+1)x}{3} = \frac{\mu^2 x^4}{4} \text{ or equivalently } x^3 = \frac{4(2\mu+1)}{3\mu^2}.$$

Since $c' = \mu x^2/2$, we write:

$$c'^3 = \frac{2(2\mu+1)^2}{9\mu}.$$

Clearly, minimising c' is equivalent to minimising c'^3 and implies:

$$\frac{8(2\mu+1)}{9\mu} = \frac{2(2\mu+1)^2}{9\mu^2}, \text{ or } 4\mu = 1+2\mu, \text{ and finally } \mu = \frac{1}{2}.$$

As a consequence $x = (32/3)^{1/3}$, $c' = x^2/4 = (16/9)^{1/3}$. Thus, the complexity of the algorithm is $L_q(1/3, 2c') = L_q(1/3, (128/9)^{1/3})$.

The case of $l_p = 2/3$. In this case, it is easier to consider a family of algorithms, indexed by t and to compute the complexity of each algorithm. We then choose $B = L_q(1/3, c')$ and $S = L_q(1/3, 2c'/(t+1))$, the total size of the sieving space and the runtime of the linear algebra are $L_q(1/3, 2c')$, the product of the norms is $L_q(2/3, 4c'/(t+1)c + tc)$ and the probability of smoothness is $L_q(1/3, (-1/3)\cdot(4/(t+1)c + tc/c'))$. As usual, we equalise c' with the constant in the probability expression and find:

$$c' = \frac{1}{3}\left(\frac{4}{(t+1)c} + \frac{tc}{c'}\right).$$

This implies:

$$3c'^2 - \frac{4c'}{(t+1)c} - tc = 0.$$

Thus:

$$c' = \frac{1}{3}\left(\frac{2}{(t+1)c} + \sqrt{\frac{4}{(t+1)^2c^2} + 3tc}\right).$$

Note that for $t = 1$, we recover the formula of the basic case. This comes to a minimum when:

$$\begin{aligned}
\tfrac{2}{(t+1)c^2} &= \tfrac{-8(t+1)^{-2}c^{-3}+3t}{2\sqrt{4(t+1)^{-2}c^{-2}+3tc}} \\
&\Leftrightarrow \\
64(t+1)^{-2}c^{-2} + 48tc &= \left(-8(t+1)^{-1}c^{-1} + 3t(t+1)c^2\right)^2 \\
&\Leftrightarrow \\
48tc &= -48tc + 9t^2(t+1)^2c^4 \ .
\end{aligned}$$

Thus, we take:

$$c = 2\left(\frac{4}{3t(t+1)^2}\right)^{1/3}.$$

As a consequence:

$$c' = \frac{4}{3}\left(\frac{3t}{4(t+1)}\right)^{1/3}.$$

We recover the basic complexity $L_q(1/3,(64/9)^{1/3})$ when $t=1$ and find the expected limit

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

when t tends to infinity. We show how the complexity constant varies with c in Figure 1.

A.3 Algorithm for Larger p

For larger p, as for smaller p, two cases arise. Either p is written $L_q(2/3,c)$ or it is of the form $L_q(l_p,c)$ with $l_p>2/3$. In the first case, the overall complexity is $L_q(1/3)$ with a constant which depends on c, in the second case, the complexity is $L_q(1/3,(64/9)^{1/3})$. The parameters in this case are the smoothness bound B and the degree D. The degree of f_1 is n, the bound on the coefficients when sieving is $S=B$ and the size of the coefficients of both f_1 and f_2, according to the construction of polynomials described above is $R=p^{n/(D+1)}=q^{1/(D+1)}$. Choose:

$$D=c_D\left(\frac{\log q}{\log\log q}\right)^{1/3}$$

and write $B=L_q(1/3,c')$. Bound the norms on the f_1 side by $RS^{n+o(1)}$ and the norms on the f_2 side by $RS^{D+1+o(1)}$. Thus the product of the norms is bounded by $B^{n+D+1+o(1)}q^{2/(D+1)}$. When $l_p>2/3$, n is negligible compared to D and the product of the norms can be rewritten as $L_q(2/3,c'c_D+2/c_D)$. This is minimised when $c_D=(2/c')^{1/2}$, and becomes $L_q(2/3,2\sqrt{2c'})$. The probability of smoothness is $L_q(1/3,-(1/3)\cdot 2\sqrt{2/c'})$. As usual, we equalise the (opposite of) this constant with c'. This yields $c'=(8/9)^{1/3}$ and finally the expected $L_q(1/3,(64/9)^{1/3})$ complexity.

When $l_p=2/3$, matters are more complicated. The product of the norms is now rewritten as $L_q(2/3,c'(c_D+1/c)+2/c_D)$, again this is minimised at $c_D=(2/c')^{1/2}$. However, it now becomes, $L_q(2/3,c'/c+2\sqrt{2c'})$. The probability of smoothness is $L_q(1/3,-(1/3)\cdot(1/c+2\sqrt{2/c'}))$. Equating this constant to c' yields:

$$(3c'-1/c)^2=8/c' \quad \text{or} \quad 9c'^3-\frac{6}{c}c'^2+\frac{1}{c^2}c'-8=0.$$

When c tends to infinity, we recover the $(64/9)^{1/3}$ constant in the complexity.