Algorithms of discrete logarithm in finite fields

Razvan Barbulescu

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Algorithmes de logarithmes discrets dans les corps finis

THÈSE

présentée et soutenue publiquement le 5 décembre 2013

pour l'obtention du

Doctorat de l'Université de Lorraine
(mention informatique)

par

Razvan Barbulescu

Composition du jury

Rapporteurs :	Jean-Marc Couveignes	Prof. Univers. Bordeaux
	Alfred Menezes	Prof. Univers. Waterloo, Canada
Examinateurs :	Nicolas Brisebarre	CR CNRS
	Emmanuel Jeandel	Prof. Univers. Lorraine
	Antoine Joux	Prof. Univers. Paris 6
	François Morain	Prof. École Polytechnique
	Frederik Vercauteren	KU Leuven, België
Directeur :	Pierrick Gaudry	DR CNRS

Laboratoire Lorrain de Recherche en Informatique et ses Applications

Contents

Introduction

Motivation

Cryptography is concerned with secure communications between two entities, for example two computers connected to the Internet. The easiest method requires that the two entities have previously exchanged a secret key which allows them to encrypt and then to decrypt the message. When this is not possible, one uses a method which allows the two entities to agree on a common secret key while using an insecure channel. This idea, which goes back to Diffie and Hellman in 1976 is the basis of public key cryptography. Its main tool, the one-way functions, are mathematical functions which are easy to compute in one direction and hard in the other. The first example [DH76] of such a function was the exponentiation of an integer modulo a prime because its inverse function, called the discrete logarithm, seemed to take an exponential time. It is the difficulty of this problem that we analyze in this thesis.

The discrete logarithm is currently in use, but other one-way functions are more popular. The most noticeable are the multiplication of integers [RSA78], having as inverse the integer factorization, and the scalar multiplication on an elliptic curve [Mil86, Kob87], whose inverse is called the elliptic curve discrete logarithm. An unexplained fact is that every time an algorithmic improvement has been made in the discrete logarithm problem (DLP) it has been translated to the factoring problem and vice-versa, making it interesting to tackle the two problems together. The case of elliptic curve cryptography is different because, except for some weak cases, the best algorithms known are exponential. Nevertheless it is also a motivation for us because, the two types of attacks on elliptic curves are either inspired from discrete logarithm algorithms [GHS02] or consist in reducing the elliptic curve discrete logarithm to the classical problem of discrete logarithm in finite fields [MOV93, FR94]. For example one can solve the DLP on super-singular elliptic curves defined over $\mathbb{F}_{2^n}$ and $\mathbb{F}_{3^n}$ if one computes discrete logarithms in $\mathbb{F}_{2^{4\cdot n}}$ and, respectively, $\mathbb{F}_{3^{6\cdot n}}$.

When comparing various algorithms for the DLP it is convenient to define the following notation:

$$L_x(\alpha, c) = \exp\left(c(\log x)^{\alpha}(\log\log x)^{1-\alpha}\right),$$

where $0 \leq \alpha \leq 1$ and $c > 0$. For example the exponential algorithms take a time $L_x(1, c)$ for some constant c. On the other hand polynomial algorithms of complexity $(\log x)^k$ can be written as $L_x(0, k)$.

Computing discrete logarithms in a prime field $\mathbb{F}_p$ has already been addressed by Kraitchik in [Kra22]. The key notion was that of smooth numbers: an integer is B-smooth if all its prime divisors smaller than B. This notion was then used in the world of factorization where Lehmer and Powers [LP31] proposed a method which proved to be very effective in [MB75]. Its complexity was $L_N(1/2, c)$, $c > 0$, but the tools to prove it were only designed a couple of years later [CEP83]. The best algorithm for discrete logarithm available in the literature in 1976 was Shanks' baby-step-giant-step of complexity $\sqrt{p} = L_p(1, \frac{1}{2})$, as it is noted in [DH76].

The first sub-exponential algorithm for discrete logarithms, Index Calculus, is inspired from Kraitchik's method and was independently discovered by two

teams [Adl79] and [Poh77]. It has a complexity of type $L_p(1/2, \cdot)$, which is as fast as the factorization algorithms known at that time. We start by choosing an integer $B > 0$ and by making a list of all the primes less than B, whose set is called the factor base. The first stage of the algorithm consists in picking random numbers from a list and testing their B-smoothness, until one collect B numbers. We will see how each smooth number produces a linear equation among the discrete logarithms of the primes in the factor base. A second step consists in solving a large linear system, which gives us the discrete logarithms of the factor base elements. A third stage called individual logarithm stage expresses the desired logarithm with respect to the discrete logarithms of the factor base elements.

Many algorithms which followed have the same main stages, in particular the modern algorithms of complexity $L(1/3, \cdot)$. In these algorithms one has an additional stage which selects two appropriate polynomials f and g in $\mathbb{Q}[x]$ or $\mathbb{F}_q[t][x]$ according to the type of finite fields we are interested in: large or small characteristic. The four stages to keep in mind are then:

- Polynomial selection. Not present in all the algorithms, this stage corresponds to the selection of two polynomials f and g subject to a set of conditions.

- Relation collection. We collect a list of numbers, respectively, of polynomials which are smooth. For example we collect pairs of integers (a, b) such that $F(a, b)$ and $G(a, b)$ are B-smooth where $F(x, y) = y^{\deg f} f(x/y)$ and $G(x, y) = y^{\deg g} g(x/y)$.

- Linear algebra stage. We solve a large system of linear equations with coefficients modulo the cardinality of the discrete logarithm group.

- Individual logarithm. We use the previous result to compute the desired logarithm. Note that we do not need to repeat the previous stages if more than one discrete logarithm is needed.

In algorithms of $L(1/3)$ complexity [Gor93, Sch93, JL03, JLSV06], the smoothness tests continue to play an important role. On the one hand, although the theoretical description of the algorithm uses a technique called sieving, in practice due to the memory constraints one proceeds in two steps: one collects pairs which are likely to be smooth and then one tests their smoothness.

The fastest smoothness test known today is Lenstra's [Len87] elliptic curve method of factorization (ECM). This heuristic algorithm has a proven counterpart using hyperelliptic curves (HECM), but ECM seems to have better performancies in practice [**?**]. ECM has been the object of many improvements, the most noticeable being the Stage 2 continuation, the curve arithmetic acceleration and the selection of curves with larger torsion over $\mathbb{Q}$. Hence it was proposed to put the elliptic curves in new forms so that the curve arithmetic needs fewer field operations [Mon92],[BL07, BBLP13]. A different direction of improvement was to select curves with better success probabilities [Suy85], [AM93], [BBL10]. This motivated us to find a unified technique which selects good curves on any family of elliptic curves and then to demonstrate its efficiency with concrete examples.

Let us return to the main topic of discrete logarithms. One tackles the fields $\mathbb{F}_p$ with p prime using the number field sieve (NFS) [LL93, Gor93, Sch93, JL03, Sch05, CS06]. Very similar is then the case of fields of small characteristic where one uses the function field sieve (FFS) [Adl94, AH99, JL02]. The intermediate case, called the middle prime case, remained less understood for over a decade. First it has been shown that FFS extends to a sharp domain of the middle prime case [JL06]. Then NFS was extended to a large domain of fields of large characteristic [JLSV06] and, in the same article, the remaining fields were attacked by a new variant of NFS. Hence we can now compute discrete logarithms in any finite field $\mathbb{F}_Q$ in time $L_Q(1/3, c)$, $c > 0$. In this thesis we have searched on the one hand for improvements in all these algorithms and on the other hand to understand the interactions between NFS and FFS.

The FFS algorithm might seem less interesting than NFS because in cryptography one has avoided the finite fields of small characteristic. Indeed, in 1984 when Index Calculus was the best algorithm for fields $\mathbb{F}_p$ with a complexity of $L_p(1/2, c)$, $c > 0$, Coppersmith published an algorithm which solves the DLP in binary fields $\mathbb{F}_Q$ in time $L_Q(1/3, c')$, $c' > 0$; they were hence much weaker than the prime fields [Cop84]. Nowadays all the finite fields as well as the factorization can be tackled with algorithms of type $L(1/3, \cdot)$. Furthermore, FFS regained importance in 2000 with the invention of pairing based cryptography, transforming the pairings from an attack into a cryptographic tool [Jou00]. In the last few years, it has been an active area of research to accelerate the FFS algorithm, with a special concern on practical efficiency [HSW+10, HSST12]. We have focused on the polynomial selection and showed that a good choice can divide the overall time by a factor of 2.

Recently, a surprising breakthrough was made by Joux in [Jou13a]. While keeping the setting of FFS, he showed that in the middle prime case one can reduce the cost of the relation collection stage using a technique called pinpointing. His idea was then applied to finite fields of very small characteristic in two independent works [Jou13b] and [GGMZ13]. It broke the barrier of the $L(1/3, \cdot)$ complexity since Joux obtained a complexity of type $L(1/4 + o(1))$. One must note that both algorithms could compute the discrete logarithms of their respective factor bases in polynomial time. Hence Joux asked if the remaining part of the algorithm, the individual logarithm stage, can be accelerated. Additionally, the new idea could not be used for all the fields in the application domain of FFS. We will give answers to these two issues by showing that, for all finite fields where FFS works, except for the middle prime case, the discrete logarithm can be computed in time $L(\alpha, \cdot)$ with $\alpha < 1/3$. In particular in the case of fields $\mathbb{F}_Q$ with $Q = q^k$ such that $q < k+2$ and $q \approx k$ one can compute the discrete logarithm in time $(\log Q)^{O(\log \log Q)}$ which in complexity theory is called quasi-polynomial, and is smaller than $L(\epsilon, \cdot)$ for any $\epsilon > 0$.

Summary of contributions

Parametrizations The search of elliptic curves which are best suited for ECM is an active topic in algorithmic number theory [Suy85, Mon92, AM93, BBL10,

BC10, Rab10], but the methods seem to be ad-hoc. We give a viewpoint which encompass all the previous methods and which allowed us to find new families of curves [**?**]. In particular given a curve we can measure its efficiency by computing the image of the Frobenius application, for which fast algorithms exist [Sut12]. The new families that we discovered were used by Bouvier in the GPU code of GMP-ECM, a very competitive software of factorization with ECM.

Smoothing Part of the cryptology community considers that an attacker can perform a slightly longer computation before the opening of the challenge. In this case the security of discrete logarithm cryptosystems would drop to the complexity of the individual logarithm stage. It is known that this stage is dominated by its first step, called smoothing. In our work, we improved it using a strategy in two, and then more, steps. The previously known complexity of $L_p(1/3, 1.44)$ was reduced to $L_p(1/3, 1.232)$.

DL factory The zoology of discrete logarithm algorithms for prime fields parallels that of the factorization algorithms, with one exception. This is why we proposed the DL factory by translating the idea of Coppersmith's factorization factory to our problem. The main difficulty was to show that the individual logarithm stage remains negligible with respect to the other stages, despite its increase in complexity. The specificity of the DL factory is that one can share the precomputed information for all the primes of a given bit-size. Hence, after some pre-computations of complexity $L_p(1/3, 2.007)$, the main phase of NFS for each prime takes time $L_p(1/3, 1.639)$, and the individual logarithm stage takes time $L_p(1/3, 1.232)$. The drawback is the use of a disk-space of $L_p(1/3, 1.639)$.

Polynomial selection for FFS The polynomial selection stage of modern algorithms as NFS or FFS were developed in an non-unified manner. Indeed, for FFS it was proposed to use purely inseparable polynomials [Cop84, Tho03], classical FFS polynomials [JL02, JL07], two rational side FFS [JL06] and inseparable polynomials [HSW+10, HSST12]. One can extend the list with more and more ideas and, as in the NFS case, one can introduce various functions like Murphy's α and $\mathbb{E}$ [Mur99, Bai11]. In [Bar13], instead of adding new properties, we compared the various methods to each other. We showed that the two rational side FFS offers a small set of polynomials and we gave the exact advantage of inseparable polynomials. Finally, we defined the ϵ function, mixing α and $\mathbb{E}$, which compares arbitrary separable polynomials. We gave experimental evidence that ϵ predicts the sieve efficiency of a polynomial up to a 5% error. It was rapid enough to allow the selection of polynomials for two record computations with FFS in characteristic 2 [BBD+12, BBD+13].

Quasi-polynomial algorithm The recent algorithms of Joux [Jou13b] and Granger and others [GGMZ13] had the particularity that the main phase computation can be done in polynomial time. The faster method for the individual logarithm stage, presented in [Jou13b], proceeds in three steps: a step corresponding to the smoothing in NFS, a classical descent which was already used in FFS

and a new technique using Gröbner basis. In a joint work with Gaudry, Joux and Thomé [BGJT13] we proposed an algorithm which allows to compute $\log P$ for any polynomial P as a linear combination of $\log Q_i$ for a set of polynomials Q_i of degree less than $\deg P/2$. Hence, the individual logarithm stage consists only in the computation of a descent tree. We showed that the number of nodes in the tree is quasi-polynomial and that the computations done at every step take a polynomial time.

Karatsuba-like formulae An additional contribution which goes beyond the scope of this document is the research of Karatsuba-like formulae. All the integer and multiplication algorithms have explicit stages of evaluation, multiplication and interpolation. A seminal paper of Montgomery [Mon05] showed that other unexplained formulae exists, and they are very effective in the arithmetic of finite fields [Alb11, Alb12]. A series of new formulae followed in the next years [CH07, FH07, Ose08, CÖ08, CO09, CKO09, CÖ10, CÖ11, CBH11], but virtually nothing was discovered after 2011, so that one could ask if these formulae are optimal. We started a project with Detrey, Estibals and Zimmermann which aimed to reproduce and extend the computations made by Montgomery. The outcome was different since we discovered a faster algorithm. It allowed us in [BDEZ12] to complete the exhaustive search of Montgomery and to prove the optimality of his formulae, as well as for most of the magical formulae known today. As a bonus, we found new formulae in characteristic 3 which can be used in cryptography; for example Algorithm 1 in [Est10] uses 12 products whereas we discovered a formula with 11 products. Note however that because of the quasi-polynomial the main application of our formula, the pairing-based cryptography in caracteristic 3, is no longer interesting.

Chapter organization

The thesis has two parts: one in which we study ECM as the best smoothness test today and one in which we address the general problem of computing discrete logarithms in any finite field.

In Chapter 1 we make a list of the smoothness results which will be needed througout the document. In Chapter 2 we introduce the ECM algorithm, which serves as basis for the next two chapters. In Chapter 3 we improve the algorithm itself and in Chapter 4 we put ECM at work in the individual logarithm stage of NFS.

We start the second part with Chapter 5 which presents basic algorithms, still used today to simplify the modern algorithms. In Chapter 6 we present a middlebrow description of the NFS and FFS algorithms. In the next chapter we compute the complexities of previously introduced algorithms, which gives us the occasion to introduce the DL factory. Some of the details that we skipped in the first description of NFS and FFS were inserted in Chapter 8 where we also present two new improvements. In Chapter 9 we focus on the polynomial selection stage of FFS, obtaining a new function to rank polynomials with respect to their sieve efficiency. The last chapter of the second part is independent from the rest. It

presents a quasi-polynomial algorithm in small characteristic as a result of the recent development on the problem.

Personal works

[Bar13] R. Barbulescu. Selecting polynomials for the function field sieve, 2013. Available at Cryptology ePrint Archive Report 2013/200, Accepted for publication in Math. Comp.

[**?**] R. Barbulescu, J. W. Bos, C. Bouvier, T. Kleinjung, and P. L. Montgomery. Finding ECM-friendly curves through a study of Galois properties. In *Algorithmic Number Theory–ANTS X*, pages 63–86, 2013.

[BBD+13] R. Barbulescu, C. Bouvier, J. Detrey, P. Gaudry, H. Jeljeli, E. Thomé, M. Videau, and P. Zimmermann. Discrete logarithm in GF(2^{809}) with FFS, 2013. Available at Cryptology ePrint Archive Report 2013/197, Accepted for presentation at the Public Key Cryptography 2014 conference.

[BDEZ12] R. Barbulescu, J. Detrey, N. Estibals, and P. Zimmermann. Finding optimal formulae for bilinear maps. In *Arithmetic of Finite Fields–WAIFI 2012*, volume 7369 of *Lecture Notes in Comput. Sci.*, pages 168–186. Springer, 2012.

[BGJT13] R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic, 2013. Cryptology ePrint Archive Report 2013/400, Submitted to Eurocrypt 2014.

Part I

Smoothness and ECM

Chapter 1

Smoothness Probabilities

Most of the sub-exponential discrete logarithm algorithms rely on the notion of smoothness, which is a building block that we will meet everywhere in this document. First introduced in the context of integers, smoothness is also important in the case of polynomials. This short chapter summarizes all the basic theoretical results about the smoothness probability which come from analytic number theory.
We proceed as follows. After giving the definition of smooth numbers we recall the main result of [CEP83]. We then reinterpret it using the L-notation, which is convenient for the analysis of discrete logarithm and factorization algorithms. We conclude the chapter with the analogous results for polynomials.

1.1 Smooth numbers

Many cryptographic algorithms generate random numbers below a given bound and test if all their prime factors are small. More formally, if B is an integer, we say that an integer is B-smooth if its prime factors are less than or equal to B. We are interested in evaluating the number $\psi(x, y)$ of positive integers less than or equal to x which are y-smooth, i.e., the cardinality of the following set

$$\Psi(x, y) = \left\{ n \in [1, x] \mid n \text{ is } y\text{-smooth} \right\}.$$

We next define formally the probability of a number in $[1, x]$ to be smooth:

$$P_{\text{smooth}}(x, y) = \psi(x, y)/x.$$

notation	meaning
$f = O(g)$	$\exists c > 0, x_0 > 0 \quad x \geq x_0 \Rightarrow \lvert f \rvert \leq c \lvert g \rvert$
$f = o(g)$	$\forall \epsilon > 0 \quad \exists x_\epsilon \quad x \geq x_\epsilon \Rightarrow \lvert f \rvert \leq \epsilon \lvert g \rvert$
$f = \Theta(g)$	$f = O(g)$ and $g = O(f)$
$f = \tilde{O}(g)$	$\exists k \in \mathbb{N} \qquad f = O\left((\log g)^k g\right)$.

Table 1.1: A list of notations.

The first asymptotic formula for $\psi(x,y)$ was that of Dickman in 1930 (see [HT93]) who proved that for any fixed $u > 0$

$$\lim_{x\to\infty} \psi(x, x^{1/u})/x = \rho(u),$$

where $\rho(u)$ is defined as the (unique) continuous function such that $\rho(u) = 1$ for $u \in [0,1]$ which satisfies the equation

$$u\rho'(u) = -\rho(u-1) \qquad (u > 1).$$

Note that this equation allows to estimate Dickman's rho to any precision on any fixed interval $[0,c]$. Moreover, when u is large enough, $\rho(u)$ can be approximated by u^{-u} (Corollary 1.3,[HT93]):

$$\lim_{u\to\infty} \rho(u) = u^{-u(1+o(1))}.$$

In practice, the relevant interval for u is $[1,10]$ on which u^{-u} has the good order of magnitude.

The drawback of Dickman's result is that it does not cover the case $\psi(x, x^{1/u})$ when u depends on x, e.g. $u = \sqrt{\log x}$. Canfield, Erdös and Pomerance proved a stronger result:

Theorem 1.1.1 ([CEP83]). *If $\epsilon > 0$ is fixed and $3 \le u \le (1-\epsilon)\log x/\log\log x$, then*

$$\psi(x, x^{1/u}) = x \exp\left\{-u\left(\log u + \log\log u - 1 + o(1)\right)\right\}.$$

In short $P_{\text{smooth}} = u^{-u(1+o(1))}$, where $o(1)$ is a function depending on x and u which tends to 0 uniformly when x tends to infinity. More generally, we make the list of the well known notations in Table 1.1.

In the complexity analysis of several algorithms we will estimate the probability that two numbers are simultaneously smooth. It turns out that we will obtain good formulae using that, for $x_1, x_2, y > 0$, one has the inequality

$$P_{\text{smooth}}(x_1, y) P_{\text{smooth}}(x_2, y) \ge P_{\text{smooth}}(x_1 x_2, y)^{1+o(1)}.$$

Nevertheless, a slightly longer argument shows that the equality holds.

Corollary 1.1.2. *Let x and y be such that $u = \frac{\log x}{\log y}$ satisfies the conditions in Theorem 1.1.1. Then for x_1 and x_2 such that $y \le x_1, x_2 \le x$ we have*

$$P_{\text{smooth}}(x_1, y) P_{\text{smooth}}(x_2, y) = P_{\text{smooth}}(x_1 x_2, y)^{1+o(1)}.$$

Proof. Put $u_1 = (\log x_1)/(\log y)$ and $u_2 = (\log x_2)/(\log y)$ and, without loss of generality, assume $u_2 \ge u_1$. Since $u_1, u_2 \ge 1$, the logarithms of u_1, u_2 and $u_1 + u_2$ are positive. Put $L(u_1,u_2) = u_2(\log u_2) + u_1(\log u_1)$ and $R(u_1,u_2) = (u_1+u_2)\log(u_1+u_2)$. We have to show that $L = R(1+o(1))$. On the one hand one has $L \le R$. For the second inequality we take $\epsilon > 0$ and we distinguish the cases according to which $u_1 \ge u_2/(\log u_2)$ or not.

If $u_1 \geq u_2/(\log u_2)$ then we use the fact that $\log u_1 = (1+o(1))\log u_2$. Indeed, we have $\log u_1 \geq \log u_2 - \log\log u_2$. Since $\log(u_1+u_2) \leq \log u_2 + \log 2$ we obtain

$$L(u_1,u_2) \geq (u_1+u_2)\left(\log(u_1+u_2) - \log 2 - \log\log u_2\right).$$

Hence, for large enough u_2 we have $L \geq (1-\epsilon)R$.

If $u_1 \leq u_2/(\log u_2)$ we use the fact that $u_2 \log u_2$ dominates both $L(u_1,u_2)$ and $R(u_1,u_2)$. On the one hand $u_2 \log u_2 \leq L(u_1,u_2)$. On the other hand we have $u_1+u_2 \leq u_2(1+\frac{1}{\log u_2})$, so

$$(u_1+u_2)\log(u_1+u_2) \leq u_2(\log u_2)\left(1+\frac{1}{\log u_2}\right)\left(1+\log\left(1+\frac{1}{\log u_2}\right)\right).$$

For large enough u_2 we have $R(u_1,u_2) \leq (1+\epsilon)u_2\log u_2 \leq (1+\epsilon)L(u_1,u_2)$. □

1.2 The L notation

The algorithms which have a complexity larger than polynomial and smaller than exponential are called sub-exponential. More formally, an algorithm is sub-exponential if there exists a constant $\alpha < 1$ such that, for an n-bit input, it takes a time less than $\exp(n^\alpha)$. The first idea would be to measure their complexity using the functions $\exp(n^\alpha)$ with $0 < \alpha < 1$. Nevertheless, all the algorithms of this thesis optimize their complexity when one of their parameters, a smoothness bound that we called B, satisfies an equation similar to the following:

$$B^{1+o(1)} = P_{\text{smooth}}(x,B)^{-1}.$$

One can easily test that, for no constant α, the equation above holds for $B = \exp((\log x)^\alpha)$. This leads us to introduce the functions below

$$L_x(\alpha,c) = \exp\left(c(\log x)^\alpha(\log\log x)^{(1-\alpha)}\right).$$

Note that the polynomial algorithms have a complexity $L(0,c)$ for some constant c, whereas the exponential ones have a complexity $L(1,c)$.

Due to the L notation we can give a simpler form for Theorem 1.1.1.

Corollary 1.2.1. *Let a,b,c,d be positive real numbers and suppose $a > c$. Then we have*

$$P_{smooth}\left(L_x(a,b),L_x(c,d)\right) = L_x\left(a-c,(a-c)\frac{b}{d}\right)^{-1+o(1)}.$$

Proof. Using Theorem 1.1.1 we know that the smoothness probability equals $\exp(-(1+o(1))u\log u)$ for $u = \log(L_x(a,b))/\log(L_x(c,d))$. This further gives $u = \frac{b}{d}(\log x)^{a-c}$ and the result follows. □

The complexity calculations in the following chapters will often benefit from a couple of easy formulae for the L functions.

Proposition 1.2.2. *Let (a,b) and (c,d) be two pairs of positive reals. Then we have*

$$L_{\{L_x(a,b)\}}(c,d) = L_x\left(ac, db^c a^{(1-c)}\right)^{1+o(1)}$$

$$\text{and} \qquad L_x(a,b)\cdot L_x(c,d) = \begin{cases} L_x(a,b)^{1+o(1)}, & \text{if } a > c; \\ L_x(a,b+d), & \text{if } a = c. \end{cases}$$

If in addition we assume that (a,b) is larger than (c,d) in lexicographical order, i.e., $a > c$ or we have $a = c$ and $b > d$, then we obtain

$$L_x(a,b) + L_x(c,d) = L_x(a,b)^{1+o(1)}.$$

1.3 Smooth polynomials

Many algorithms dedicated to integer arithmetic can be translated to the case of polynomials. Even more, in Chapter 6 we see two algorithms on number and respectively function fields which correspond to each other if one replaces numbers with polynomials. In particular, we extend the smoothness definition and say, for an integer β called smoothness bound, that a polynomial is β-smooth if all its irreducible factors have degree less than or equal to β. Corresponding to ψ in numbers' world, for any finite field $\mathbb{F}_q$, we denote the number of smooth polynomials by

$$N_q(n,m) = \#\left\{h(t) \in \mathbb{F}_q[t], \deg h(t) = n, \text{ monic and } m\text{-smooth}\right\}.$$

Surprisingly, the proportion $N_q(n,m)/q^n$ has a limit for any finite field $\mathbb{F}_q$, independent on q and it can be written in terms of Dickman's rho. We cite a theorem proven using Cauchy's coefficient formula.

Theorem 1.3.1 ([PGF98])**.** *The number of m-smooth monic polynomials of degree n over $\mathbb{F}_q$ satisfies*

$$N_q(n,m) = q^n \rho\left(\frac{n}{m}\right)\left(1 + O\left(\frac{\log n}{m}\right)\right),$$

where $O()$ is a function independent on q.

If one puts $u = \frac{n}{m}$ this theorem states that the smoothness probability is $u^{-u(1+o(1))}$. If one bounds oneself to a more basic technique, one can find a lower bound of u^{-cu} for an explicit constant $c > 0$.

Proposition 1.3.2 ([JL06])**.** *Let $0 < \alpha_1 < \alpha_2 < 1$ be two constants. For any finite field $\mathbb{F}_q$ and for any positive integers $m \geq 8$ and n such that $n^{\alpha_1} < m < n^{\alpha_2}$ we have*

$$N_q(n,m)/q^n \geq u^{-cu},$$

for any constant c larger than $1/(1-\alpha_2)$.

Proof. The basic idea is to estimate the number $I_m(q)$ of monic irreducible polynomials over $\mathbb{F}_q$ which have degree m:

$$I_m(q) = \frac{1}{m}\sum_{d|m}\mu(d)q^{m/d} \geq \frac{1}{m}\left(q^m - \lceil \log_2 m \rceil q^{m/2}\right),$$

where μ is Möbius' function. As $m \geq 8$ the right hand side member in the inequality above is larger than $q^m/2m$.

Since the number of irreducible polynomials of degree m is large compared to the number of other irreducible polynomials of smaller degree, a good guess is that a large number of m-smooth polynomials have $\ell := \lfloor n/m \rfloor$ irreducible factors of degree m. Let us find a lower bound for

$$T_q(n,m) := \#\{h(t) \in \mathbb{F}_q[t] \mid \deg h(t) = n \text{ monic with } \ell \text{ distinct irred. degree-}m \text{ factors}\}.$$

The value of $T_q(n,m)$ is clearly $\binom{I_m(q)}{\ell} q^{n-m\ell}$. Hence we obtain

$$\frac{T_q(n,m)}{q^n} = \frac{1}{\ell! q^n}\prod_{i=1}^{\ell}\left(I_m(q) - i\right) \geq \frac{(I_m(q)/2)^\ell}{\ell!} \geq \frac{1}{\ell!(4m)^\ell}.$$

Taking logarithms and using Stirling's formula we get $\log(T_q(n,m)/q^n) \geq -(1-\epsilon)\ell(\log\ell + \log m + \log 4)$ for any $\epsilon > 0$. Using the conditions on m and n we have $\log m + \log\ell \geq c\log\ell$ for any constant $c > 1/(1-\alpha_2)$, which completes the proof. □

Chapter 2

The Elliptic Curve Method of factorization

Testing the smoothness of an integer is a difficult task. The best known algorithm is Lenstra's ECM [Len87], which is central in this document. It finds the factors of an integer N below a bound $\mathcal{B}$ and becomes a factorization algorithm when $\mathcal{B} = \sqrt{N}$. This chapter provides the background about the algorithm, which will be used in the next two chapters. Indeed, in Chapter 3 we will propose an improvement to the algorithm itself. In Chapter 4 we will use ECM to improve an important building block of the discrete logarithm algorithms, called smoothing. We start the chapter by recalling the basics of elliptic curves. Then we present a very basic version of ECM, as a generalization of Pollard's $p-1$ algorithm, and give the analysis of its complexity. The rest of the chapter is devoted to the classical improvements of the algorithm. This is not exhaustive as we do not present the Stage 2 improvement.

2.1 Elliptic curves

We propose a presentation of elliptic curves which avoids to introduce general results from algebraic geometry. Hence, we prove the particular cases of interest for ECM by direct computations or by repeating the general arguments.

Given a field K, we denote by $\mathbb{P}^n(K)$ the n-dimensional projective space $\left(K^{n+1}\backslash(0,\ldots,0)\right) / \equiv$ where $(x_0,\ldots,x_n) \equiv (x'_0,\ldots,x'_n)$ if and only if there exists a non zero λ in K such that $(x'_0,\ldots,x'_n) = (\lambda x_0,\cdots,\lambda x_n)$. The class of $(x_0,\ldots,x_n)$ is denoted $(x_0 : \cdots : x_n)$.

Let $\overline{K}$ denote an algebraic closure of K. An algebraic set defined over K is any subset V of $\mathbb{P}^n(\overline{K})$ given as the solutions of a system of homogeneous polynomials with coefficients in K. For any field L containing K, the L-rational points of V are the zeros in $\mathbb{P}^n(L)$ of the equations which define V; we denote by $V(L)$ the set of L-rational points. If the ideal $I(V)$ of polynomials in $\overline{K}[x_0,\ldots,x_n]$ which vanish on V is prime we say that V is a variety.

A morphism between varieties V_1 and V_2 is a map $\varphi : V_1 \to V_2$ which can be expressed by a finite set of formulae. In more detail, a morphism is given by the definition of a finite set of $(n+1)$-tuples $g^{(i)}$, $i = 1, 2, \ldots$ as follows. Each $g^{(i)}$ is

defined on an open subset of V_1, i.e., on the complementary of a finite set. Each $g^{(i)}$ is given by $n+1$ homogeneous polynomials $g^{(i)} = (g_0^{(i)}, \cdots, g_n^{(i)})$ which do not vanish simultaneously on U_i. If two tuples $g^{(i)}$ and $g^{(j)}$ are defined at the same point P then $g^{(i)}(P) = g^{(j)}(P)$. Finally, at least one $g^{(i)}$ is defined at every point. If between two varieties V_1 and V_2 there exists a bijection φ such that φ and φ^{-1} are morphisms, we say that V_1 and V_2 are isomorphic.

A variety of $\mathbb{P}^2$ given by one equation is called a plane curve. If a plane curve is given by an equation $P(x, y, z) = 0$, we say that a point $(x : y : z)$ is singular if and only if $\frac{\partial P}{\partial x} = \frac{\partial P}{\partial y} = \frac{\partial P}{\partial z} = 0$. We can now define the elliptic curves.

Definition 2.1.1. Let A and B be two elements of a field K, $\mathrm{char}(K) \neq 2, 3$. Assume that the equation below has no singular point in $\mathbb{P}^2(\overline{K})$:

$$E_{\mathcal{W},A,B} : y^2 z = x^3 + Axz^2 + Bz^3. \tag{2.1}$$

Then an elliptic curve in short Weierstrass form is the set of solutions in $\mathbb{P}^2(\overline{K})$ of the above equation.

We also call elliptic curve every plane curve which is isomorphic to an elliptic curve in short Weierstrass form. We warn the reader that wider definitions of elliptic curves exist in characteristic 2 and 3, but they are not necessary for the ECM algorithm. When, for an elliptic curve E we write an isomorphic elliptic curve in short Weierstrass form we say that we "put E in short Weierstrass form".

In the sequel we will write affine equations and formulae obtained by setting $z = 1$. Nevertheless, all the formulae must be read in projective coordinates, i.e., each monomial must be multiplied by the right power of z in order to obtain homogeneous polynomials. For example the equation $y^2 = x^3 + Ax + b$ must be read as Equation (2.1).

Note now that in short Weierstrass form one tests if $E_{\mathcal{W},A,B}$ has singular points by simply testing if $\mathrm{Disc}(x^3 + Ax + B) = -(4A^3 + 27B^2)$ is zero. This quantity is called the discriminant of E and it is denoted $\Delta(E)$.

If $E_{\mathcal{W},A,B}$ is an elliptic curve over $\mathbb{Q}$ one can associate it to a curve over $\mathbb{F}_p$ for almost all primes p.

Definition 2.1.2. Let $A = A_1/A_2$ and $B = B_1/B_2$ be two rational numbers such that $E_{\mathcal{W},A,B}$ is an elliptic curve. If p is a prime which does not divide $6\Delta(E_{\mathcal{W},A,B})$ nor the denominators A_2 and B_2 of A and B we say that $E_{\mathcal{W},A,B}$ has good reduction[1] modulo p. We define $E_{\mathcal{W},A,B}(\mathbb{F}_p)$ as the elliptic curve:

$$E_{\mathcal{W},\overline{A},\overline{B}} : y^2 z = x^3 + \overline{A}x^2 z + \overline{B}z^3, \tag{2.2}$$

where $\overline{A} = (A_1 \bmod p) \times (A_2 \bmod p)^{-1}$ and $\overline{B} = (B_1 \bmod p) \times (B_2 \bmod p)^{-1}$ are the reductions of A and B modulo p.

It is now time to define the group structure of an elliptic curve. See Figure 2.1 for an illustration.

[1] This is an ad-hoc definition, which is different from the classical one using the notion of minimal model (see [Sil09, Chapter VII]).

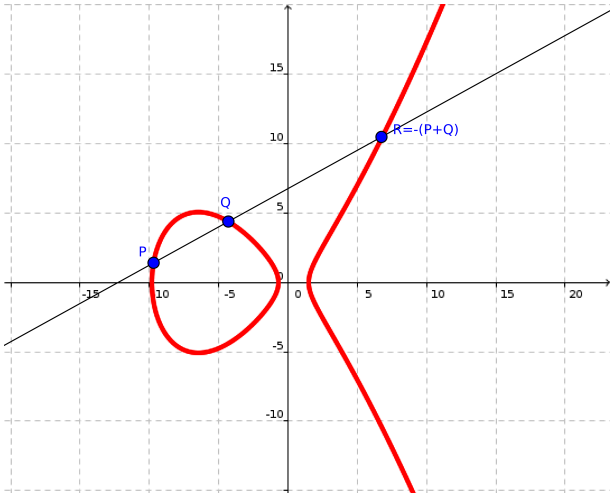


Figure 2.1: Addition law on a curve in short Weierstrass form.

Definition 2.1.3. Let $E_{\mathcal{W},A,B}$ be an elliptic curve in short Weierstrass form over a field K, $\operatorname{char} K \neq 2,3$. For any point $P = (x : y : z)$ we define its inverse by $-P = (x : -y : z)$ and then we define the law as follows.

1. For any distinct points P and Q we call $P+Q$ the inverse of the third point of intersection of the line PQ with the curve.

2. For any point P we call $P + P$ the inverse of the third point of intersection t_P with the curve, where t_P is the tangent line at P to $E_{\mathcal{W},A,B}$.

Clearly the definition associates to any two points on the curve a third one on the same curve and it is trivially commutative. Moreover, the composition law that we defined in a geometric way can also be defined in an algebraic manner. Here is the definition in affine coordinates:

$$\begin{aligned}
\lambda &= \begin{cases} \frac{y_2-y_1}{x_2-x_1} & \text{if } x_1 \neq x_2 \\ \frac{3x_1^2+A}{2y_1} & \text{if } (x_1,y_1) = (x_2,y_2) \end{cases} \\
\nu &= \begin{cases} \frac{y_1x_2-y_2x_1}{x_2-x_1} & \text{if } x_1 \neq x_2 \\ \frac{-x_1^2+Ax_1+2B}{2y_1} & \text{if } (x_1,y_1) = (x_2,y_2) \end{cases} \\
x_3 &= \lambda^2 - (x_1+x_2) \\
y_3 &= -\lambda x_3 - \nu.
\end{aligned}$$

Theorem 2.1.4 (Chapter III,[Sil09]). *Let E be an elliptic curve in Weierstrass form. Then we have:*

(i) Each of the coordinates of the sum point $P+Q$ can be expressed as a polynomial in the coordinates of P and Q.

(ii) The law endows E with a group structure.

(iii) The unique point $(x : y : z)$ of E such that $z = 0$ is the neutral element $(0 : 1 : 0)$.

In the ECM algorithm we use in an important manner an estimate on the number of points of an elliptic curve over a finite field $\mathbb{F}_q$, as given below.

Theorem 2.1.5 (Hasse). *Let $E/\mathbb{F}_q$ be an elliptic curve defined over the field with q elements. Then*

$$\left|\#E(\mathbb{F}_q) - q - 1\right| \leq 2\sqrt{q}. \tag{2.3}$$

2.2 The ECM algorithm

The subroutine of ECM can be understood by analogy with the "$p-1$" method. This latter searches the factors p of an integer N such that $p-1$ is B_1-smooth for a parameter B_1. The basic idea is that, if x_0 is coprime to N, and hence to p, and if M is a multiple of $p-1$ then p divides $x_0^M - 1$. Hence we write Algorithm 2.1.

Algorithm 2.1 The "$p-1$" method

1: Pick random $x_0 \in [2, N-2]$ coprime to N.
2: $M \leftarrow \prod_{q \leq B_1, \text{ prime}} q^{\lfloor \log_q N \rfloor}$.
3: Evaluate $x_0^M \bmod N$.
4: $g \leftarrow \gcd(x_0^M - 1, N)$.
5: **if** $g > 1$ **then**
6: **return** g;
7: **else**
8: **return** "FAIL".
9: **end if**

Note that the algorithm can be executed since we only used N, not its to-be-found factor p. If p is such that $p-1$ is B_1-smooth, then the parameter M set at line 2. is a multiple of $p-1$. Hence g is a multiple of p so the algorithm outputs a multiple of p. On the contrary assume that p' is a prime factor of N such that M is not a multiple of $p'-1$. Then $p'-1$ has a large prime factor ℓ which is coprime to M. Hence $x_0^M \not\equiv 1 \mod p'$ except if the order of x_0 is coprime to ℓ, which happens with probability $1/\ell$. We conclude that g is a proper factor of N except if we made an unlucky choice of x_0 or if all the prime divisors of N are such that $p-1$ is B_1-smooth. For most inputs N this is not a problem because we can run the algorithm with a smaller parameter B_1 until we manage to distinguish between the factors of N. Among the bad cases we have $N = F_3 \times F_4$ where $F_n = 2^{2^n} + 1$.

The building block of ECM is a variation of the "$p-1$" method where the multiplicative group $(\mathbb{Z}/p\mathbb{Z})^*$ is replaced with the group of points of an elliptic curve. Let E be an elliptic curve with integer coefficients, $E : y^2 = x^3 + Ax + B$, and let x_0, y_0, z_0 be three integers such that $P_0 = (x_0 : y_0 : z_0)$ is a point of $E(\mathbb{Q})$. We further assume that the order of P_0 is infinite. Recall that the addition formulae can be expressed by polynomial formulae with integer coefficients. Hence, if the starting point has integer coordinates, all its multiples have integer coordinates. We say that we compute a point of $E(\mathbb{Q})$ modulo N if we compute the residues modulo N of its three coordinates. Hence computing modulo N means that all the intermediate results have coordinates in $[0, N-1]$.

Note that, if p is a prime divisor of N, then if one further reduces modulo p the coordinates modulo N, then one obtains the coordinates in $E(\mathbb{F}_p)$. Some subtleties reside in the fact that two points can be equal modulo p but different modulo N. In this case one mistakenly uses the addition formulae specific to pairs of distinct points when adding two equal points. This is easy to avoid because one can compute the gcd of the two points' coordinates and find an "accidental" factor of N. Alternatively, one can check that if one uses the mistaken formula in ECM then one obtains the forbidden triplet $(0,0,0)$ in $\mathbb{F}_p^3$. Further, all the computations which follow will remain congruent to $(0,0,0)$ modulo p, so we can still compute p as the greatest common divisor of N and one coordinate. We can now present Algorithm 2.2.

Algorithm 2.2 Building block of ECM

Input: a curve $E_{\mathcal{W},A,B}$ over $\mathbb{Q}$, a point P_0 of infinite order and a parameter B_1.

1: $M \leftarrow \prod_{q \leq B_1 \text{ prime}} q^{\lfloor \log_q N \rfloor}$
2: Evaluate $(X : Y : Z) = M \times P_0$ modulo N
3: $g \leftarrow \gcd(Z, N)$
4: **if** $g > 1$ **then**
5: **return** g;
6: **else**
7: **return** "FAIL".
8: **end if**

As in the case of the "$p-1$" method, let us suppose that N has a factor p such that the order of $E(\mathbb{F}_p)$ is B_1-smooth. Then M is a multiple of $\#E(\mathbb{F}_p)$ and $M \times P_0$ is the neutral element. By Theorem 2.1.4 this shows that $Z \equiv 0 \mod p$, so g is a multiple of p. If on the contrary, p' is a prime such that the order of P_0 is not a divisor of M, then $g \not\equiv 0 \mod p'$. Also note that the algorithm fails if E has bad reduction at the divisors of N, but this can be tested by computing $\gcd(N, \Delta(E))$. Hence, except for an unlucky choice of E and P_0, the output g is a proper divisor of N.

The reader might wonder what is the advantage of Algorithm 2.2 when compared to Algorithm 2.1. If one runs them as complete algorithms, with the parameter B_1 equal to the bound $\mathcal{B}$ up to which we search for factors, then they are very similar. Nevertheless, H. Lenstra proposed to run Algorithm 2.2 as subroutine and to set parameter B_1 to a much smaller value than $\mathcal{B}$; one tests as many curves E as needed until we find an elliptic curve such that $\#E(\mathbb{F}_p)$ is B_1-smooth. Indeed, a theorem of H. Lenstra states that all the values in a subset of the Hasse interval, given by Theorem 2.1.5, correspond to some elliptic curves $E_{\mathcal{W},A,B}$ and that the distribution of these values is sufficiently uniform. Hence, each run of Algorithm 2.2 corresponds to rolling a dice.

Before passing to the analysis, note that the algorithm above is called the Stage 1 of ECM. It was later proposed to search for curves E such that $\#E(\mathbb{F}_p)$ has all its prime divisors below the bound B_1 except for one factor π which is below a second bound B_2. For this, one can use the point $P = (X : Y : Z)$ computed at step 2. of Algorithm 2.2. Next, one computes all the values $\pi \times P$ for the primes

Algorithm 2.3 The Elliptic Curve Method of factorization: stage 1.

Input: an integer N and a smoothness bound $\mathcal{B}$
$B_1 \leftarrow L_{\mathcal{B}}(1/2, \sqrt{2})$
while no factor found **do**
Pick random A and $B \in \mathbb{Z}$ and a point P_0 on $E_{\mathcal{W},A,B,/}\mathbb{Q}$ of infinite order
Run Algorithm 2.2 on N for parameter B_1, curve $E_{\mathcal{W},A,B}$ and point P_0
end while

$\pi \in [B_1, B_2]$ by a meet-in-the-middle technique. Since this development, called Stage 2, is independent of Stage 1 we refer the reader to [Mon92].

2.2.1 Complexity analysis

The cost of running one curve Let us first compute the complexity of Algorithm 2.2. The evaluation of M in line 1. can be done in B_1 stages of $(\log B_1)^{O(1)}(\log N)^2$ operations each, whereas in practice it is a pre-computation. The step in line 2. can be done by a Fast Exponentiation technique, corresponding to $O(\log M)$ operations on the curve, modulo N. As an upper bound for $\log M$ we have

$$\log M \leq \sum_{q \leq B_1} \log_q N \log q = O\left(\log N \cdot B_1\right).$$

Hence, Algorithm 2.2 has a cost of $B_1(\log N)^3$ bit operations.

The success probability We call $P_{\mathcal{B}}(B_1)$ the probability that we find a factor below $\mathcal{B}$ after running a random curve with parameter B_1. In order to estimate $P_{\mathcal{B}}(B_1)$, let us consider the set S of the B_1-smooth elements in the interval $(p - \sqrt{p}, p + \sqrt{p})$ which is included in the Hasse interval. The following theorem states that, up to a factor $O(\log p)$, the probability that a random curve has its cardinality in S is at least $\#S/(2\sqrt{p})$.

Theorem 2.2.1 ([Len87]). *There is a positive effective computable constant c such that for every prime number $p > 3$ and every set S of integers s with $|s-(p+1)| \leq \sqrt{p}$, the number of triples $(A, x, y) \in \mathbb{F}_p^3$ for which*

$$\Delta(E_{\mathcal{W},A,B}) \neq 0 \text{ and } \#E_{\mathcal{W},A,B}(\mathbb{F}_p) \in S$$

where $B = y^2 - x^3 - Ax$, is at least $\frac{cp^3}{\log p}\frac{\#S-2}{2\sqrt{p}}$.

Hence, we succeed with probability $\frac{c}{\log p} \cdot \frac{\#S}{2\sqrt{p}}$, which can be simplified to $\#S/(2\sqrt{p})$ if one forgets the $\log p$ factors in the complexity of the algorithm. We have then to estimate the probability that a number in the interval $(p-\sqrt{p}, p+\sqrt{p})$ is B_1-smooth. This is a particular case of the problem of the Smooth Numbers in Short Interval [Gra08].

Problem 2.2.2. *Let x, y and z be positive numbers such that $z, y < x$. Decide if, except for a negligible set of triples, we have*

$$\frac{\psi(x+z, y) - \psi(x, y)}{z} = P_{\text{smooth}}(x, y)^{1+o(1)}. \tag{2.4}$$

Among the answers given, the one in [HT93] covers the case where $y \geq L(5/6+o(1),1)$ and $z = y \cdot L(1/6+o(1),1)$, but we will see that this is not enough for proving the complexity of ECM.

We make the heuristic assumption that the elements in the interval $(p-\sqrt{p}, p+\sqrt{p})$ have the same probability to be B_1-smooth as the elements in $[1,p]$. Hence, the success probability of each curve is as follows

$$P_{\mathcal{B}}(B_1) \geq P_{\text{smooth}}(\mathcal{B}, B_1)^{1+o(1)}. \tag{2.5}$$

Optimal parameters Since each curve run in ECM is an independent trial, if N has a factor p below $\mathcal{B}$, then one has a probability $1/2$ to find p after testing $O(1/P_{\mathcal{B}}(B_1))$ curves. The cost of running a curve (Algorithm 2.2) is $B_1(\log N)^3$, so the total cost is

$$\text{cost(ECM)} \geq \frac{B_1(\log N)^3}{P_{\text{smooth}}(\mathcal{B}, B_1)}. \tag{2.6}$$

Note that a small bound B_1 means a large number of trials before success. Also, a large parameter B_1 means a large cost of each curve. As an intermediate value we set $B_1 = L_{\mathcal{B}}(\alpha, c)$ with $\alpha \in [0,1]$ and $c > 0$, that we will show to be optimal when set as in Algorithm 2.2. By Corollary 1.2.1, $P_{\text{smooth}}(\mathcal{B}; B_1) = L_{\mathcal{B}}(1-\alpha, (1-\alpha)/c)^{1+o(1)}$. Therefore the overall complexity is

$$L_{\mathcal{B}}\left(\max(\alpha, 1-\alpha), O(1)\right)(\log N)^3$$

as given by the formulae in Proposition 1.2.2. The optimal value is $\alpha = 1/2$. Further, the complexity becomes $L_{\mathcal{B}}(1/2, c')^{1+o(1)}$ for

$$c' = c + \frac{1}{2c}.$$

We minimize this by setting $c = 1/\sqrt{2}$, so that $c' = \sqrt{2}$. We conclude with the following fact which presents ECM as a Monte Carlo algorithm.

Fact 2.2.3. *Let N be an integer and let $\mathcal{B}$ be a bound. We assume that the probability that an integer in the interval $(p-\sqrt{p}, p+\sqrt{p})$ is $L_p(1/2, \sqrt{2})$-smooth with the same probability as an integer in the interval $(1,p)$, up to an $1+o(1)$ exponent. If N has a prime factor below $\mathcal{B}$, then we have a probability of $1/2$ to find it with ECM after at most $L_{\mathcal{B}}(1/2, \sqrt{2})^{1+o(1)}(\log N)^3$ operations.*

2.3 Classical improvements

The Stage 1 of ECM was improved in two ways: by speeding up the curve arithmetic and by restricting to elliptic curves with a higher probability of success. Each of these directions are followed in this section.

2.3.1 Arithmetic

Montgomery parametrization Given an elliptic curve E in Weierstrass form, for each value of x, there are at most two values of y such that $(x:y:1)$ is on

E. Hence, when one computes the sum of two points P and Q on E, it seems expensive to compute the y coordinate since it only allows to distinguish between two points $P+Q$ and $-(P+Q)$.

Let us call x_+ and x_- the x coordinates of the sum and the difference of two distinct points $P_1 = (x_1 : y_1 : 1)$ and $P_2 = (x_2 : y_2 : 1)$. Montgomery noted that x_+ can be computed from x_1, x_2 and x_-. To see this, let $E : y^2 = b_3x^3+b_2x^2+b_1x+b_0$ be an elliptic curve. Then an easy computation gives the formulae for x_+ and x_- below

$$\begin{aligned} x_+ &= \frac{\left(\frac{y_1-y_2}{x_1-x_2}\right)^2 - b_2}{b_3} - x_1 - x_2 \\ x_- &= \frac{\left(\frac{y_1+y_2}{x_1-x_2}\right)^2 - b_2}{b_3} - x_1 - x_2. \end{aligned}$$

Hence, if we put

$$P(x) = ((x + x_1 + x_2)b_3 + b_2)(x_1 - x_2)^2,$$

then $P(x_+) = (y_1 - y_2)^2$ and $P(x_-) = (y_1 + y_2)^2$, so $P(x_+)P(x_-) = (y_1^2 - y_2^2)^2$. This is an expression in x_1 and x_2 alone when using the equation of E.

The second idea is to chose values for the parameters b_0, b_1, b_2, b_3 which simplify the formulae. In his thesis, Montgomery writes "but ECM lets its user to choose the curve, and there is no prohibition against selecting one of the form [(2.7)]".

Definition 2.3.1. Let A and B be two elements of a non-binary field K. Assume that $(A^2 - 4)B$ is non zero. A Montgomery curve is an elliptic curve with affine equation:

$$E_{\mathcal{M},A,B} : By^2 = x^3 + Ax^2 + x. \tag{2.7}$$

Note that it corresponds to setting $b_1 = 1/B^2$, $b_2 = A/B$, $b_3 = 1$, $b_2 = A$ and $b_0 = 0$ together with making the change in coordinates $x = x'/B$ and $y = y'/B^2$. This simplifies the expression of x_+ as follows

$$x_+ = \frac{1}{x_-}\left(\frac{x_1x_2 - 1}{x_1 - x_2}\right)^2. \tag{2.8}$$

One can therefore compute in affine coordinates using x only or, in projective coordinates, using $(X : Z)$ only.

Since, in order to compute $P + Q$ one needs $P - Q$, the Fast Exponentiation algorithm is replaced by a series of computations of the form $m \times P$ where m are elements in a chain as follows. A Lucas chain is any sequence of integers such that $i+j$ can occur only if $|i-j|$ does. For example any sequence $(a_n)_{n\in\mathbb{N}}$ which satisfies $a_{n+2} = a_{n+1} + a_n$ is a Lucas chain. Montgomery proposed the PRAC algorithm which constructs a Lucas chain for any positive integer a and which heuristically reduces the number of intermediate values. The idea is to choose a value b less than a and then to iterate the operation $(a, b) \to (b, a - b)$ until one obtains $b = 1$. It is known that such sequences have asymptotically $\log_\phi a$ elements where $\phi = \frac{(1+\sqrt{5})}{2}$

is the golden ratio. PRAC minimizes this number in practice by setting b close to $\lfloor a/\phi \rfloor$.

This short presentation of the Montgomery curves in ECM omitted many issues. The reader can find a good review in [ZD06] and can experiment with ECM using GMP-ECM [Z+13].

The Edwards form Euler and Gauss studied the solutions in $\mathbb{R}$ of the equation $x^2+y^2+x^2y^2 = 1$ and noted what in modern language corresponds to a group law on this set. This led Edwards [Edw07] to study the equations of type $x^2 + y^2 = a^2(1+x^2y^2)$ where a is any element of a field K. Finally, Bernstein et al. [BBJ+08] defined the twisted Edwards curves as follows.

Definition 2.3.2. Let K be a non-binary field, i.e., $\mathrm{char}(K) \neq 2$. Let a and d be two distinct non-zero elements of K. We call twisted Edwards curve the solutions in $\mathbb{P}^2(\overline{K})$ of the affine equation

$$E_{\mathcal{E},a,d} : ax^2 + y^2 = 1 + dx^2y^2. \tag{2.9}$$

The link between the twisted Edwards and Weierstrass curves will be studied later in this section. Although an Edwards curve is a twisted Edwards curve $E_{\mathcal{E},a,d}$ with $a = 1$, in this work we abusively drop the word "twisted" when talking of any twisted Edwards curve. Also note that the twisted Edwards curves with $a = -1$ play an important role, hence we denote the curve $E_{\mathcal{E},-1,d}$ by E_d.

Theorem 2.3.3 ([BBJ+08]). *Let K be a non-binary field and $E_{\mathcal{E},a,d}$ a twisted Edwards curve. Assume that both d/a and d are non-squares in K. Then the following formulae define a group law on $E_{\mathcal{E},a,d}(K)$ for which $(0, 1)$ is the neutral element.*

$$(X, Y) = \left(\frac{xy' + ax'y}{1 + dxx'yy'}, \frac{yy' - axx'}{1 - dxx'yy'} \right) \tag{2.10}$$

The proof is straight-forward and can be done using a computer algebra software package.

The unified law One advantage of the Edwards curves is that, if d and d/a are non-squares, the addition law is the same when one computes $P + Q$ for distinct or equal points P and Q. In public-key protocols, the cryptographer can select a curve $E_{\mathcal{E},a,d}$ with $a = 1$ and a base field $\mathbb{F}_p$ such that d is not a quadratic residue modulo p. This is not the case for ECM as it is not possible to know if an integer N has a factor such that a parameter d is a quadratic non-residue modulo p (a Jacobi symbol -1 modulo N does not guarantee a Legendre symbol -1 for all factors of N). We next explain how to circumvent this difficulty.

Note first that if one considers a point $P = (x_0, y_0)$ on an Edwards curve $E_{\mathcal{E},a,d}$ with $a \neq 1$ one obtains

$$(x_0, y_0) + \left(\frac{1}{\sqrt{d}} \frac{1}{y_0}, -\frac{1}{\sqrt{d}} \frac{1}{x_0} \right) = (\infty, c), \tag{2.11}$$

where c is a non-zero rational fraction in x_0 and y_0. Therefore we have to add some points at infinity in order to have a group structure on $E_{\mathcal{E},a,d}(L)$ in fields L where d is a square.

Definition 2.3.4. Let $a, d \in K$ be such that $ad(a-d) \neq 0$. A completed Edwards curve is the set

$$E_{\overline{\mathcal{E}},a,d} = \left\{ \big((X:Z),(Y:T)\big) \in \mathbb{P}^1(\overline{K}) \times \mathbb{P}^1(\overline{K}) \mid aX^2T^2 + Y^2Z^2 = Z^2T^2 + dX^2Y^2 \right\}. \tag{2.12}$$

We can now homogenize the law in Theorem 2.3.3 by setting $x = X/Z$ and $y = Y/T$, which we call Edwards formulae too. We obtain the following result.

Proposition 2.3.5 (Theorem 8.1, [BL11])**.** *Let $E_{\overline{\mathcal{E}},a,d}/K$ be a completed Edwards curve. Then we have*

(i) The non-affine points of $E_{\overline{\mathcal{E}},a,d}$ are $M_\pm = (\infty, \pm\sqrt{\frac{a}{d}})$ and $N_\pm = (\pm\frac{1}{\sqrt{d}}, \infty)$.

(ii) M_- and M_+ have order 2 whereas N_- and N_+ have order 4.

(iii) The only case where the addition formulae for $P+Q$ are invalid, i.e., produce $(X,Z) = (0,0)$ or $(Y,T) = (0,0)$, is when $P-Q$ belongs to $\{M_-, M_+, N_-, N_+\}$.

The case when the Edwards formulae fail is handled with the following set of formulae given in [HWCD08]:

$$\begin{aligned} X_3 &= X_1Y_1Z_2T_2 + X_2Y_2Z_1T_1 \\ Z_3 &= aX_1X_2T_1T_2 + Y_1Y_2Z_1Z_2 \\ Y_3 &= X_1Y_1Z_2T_2 - X_2Y_2Z_1T_1 \\ T_3 &= X_1Y_2Z_2T_1 - X_2Y_1Z_1T_2. \end{aligned}$$

Hence, when running ECM on an Edwards curve, $M \times P_0$ can be evaluated as $i \times (2^m \times P_0)$ where i and m are integers such that $M = 2^m i$ and i is odd. After the evaluation of $2^m \times P_0$, all the computations take place in a subgroup of odd order. Hence, according to Proposition 2.3.5, they can be done using exclusively the Edwards formulae.

Equivalence of Edwards and Montgomery curves The Montgomery curves were studied several years before the introduction of the Edwards curves. Hence, despite a better arithmetic, the Edwards had at the moment of their publication the disadvantage that no good families of Edwards curves had good torsion properties. Fortunately, any twisted Edwards curve can be transformed into a Montgomery curve and vice-versa.

To see this, we put an Edwards curve into a Weierstrass form. Put $x := u/v$ and $y := (u-1)/(u+1)$ for variables u and v. Then the equation of $E_{\mathcal{E},a,d}$ becomes

$$\begin{aligned} & au^2(u+1)^2 + (u-1)^2v^2 = v^2(u+1)^2 + du^2(u-1)^2 \\ \Leftrightarrow\ & 4uv^2 = u^2[((a-d)u^2 + (a+d)u + (a-d)] \\ \Leftrightarrow\ & v^2 = \alpha u^3 + \beta u^2 + \gamma u, \end{aligned}$$

for parameters α, β, γ, where we could divide by 4 as the field is non-binary. This brings us to the main result of this paragraph.

Theorem 2.3.6 (Theorem 7.1,[BL11]). *Let a, d, A and B be four elements of a non-binary field such that $ad(a-d) \neq 0$ and*

$$A = 2\frac{a+d}{a-d} \text{ and } B = \frac{4}{a-d}. \tag{2.13}$$

Then $E_{\overline{\mathcal{E}},a,d}$ and $E_{\mathcal{M},A,B}$ are birationally equivalent over K, i.e., there is a bijection φ between $E_{\overline{\mathcal{E}},a,d}$ and $E_{\mathcal{M},A,B}$ such that, except for a finite set of points, both φ and φ^{-1} are rational functions with coefficients in K.

Proof. Let us define φ as follows.

$$\begin{array}{rcl} E_{\overline{\mathcal{E}},a,d} & \to & E_{\mathcal{M},A,B} \\ (X:Z,Y:T) & \mapsto & \big((T+Y)X:(T+Y)Z:(T-Y)X\big) \\ (0:1,-1:1) & \mapsto & (0:0:1) \end{array}$$

A straightforward computation shows that its inverse φ^{-1} is as follows:

$$\begin{array}{rcl} E_{\mathcal{M},A,B} & \to & E_{\overline{\mathcal{E}},a,d} \\ (u:v:w) & \mapsto & \big(u:v,(u-v):(u+v)\big) \\ (0:0:1) & \mapsto & (0:1,-1:1) \\ (0:1:0) & \mapsto & (0:1,1:1) \end{array}$$

□

Note that the theorem states in elementary language that the curves $E_{\mathcal{E},a,d}$ and $E_{\mathcal{M},A,B}$ are isomorphic. A direct consequence is that, for any field L, $E_{\overline{\mathcal{E}},a,d}$ and $E_{\mathcal{M},A,B}$ have the same number of rational points over L. In particular, except for a small number of accidental factors found during the intermediate computations, running ECM with a Montgomery curve succeeds if and only if ECM succeeds on its corresponding twisted Edwards curve.

Corollary 2.3.7. *Let a, d, A and B be four rational numbers such that $ad(a-d) \neq 0$, $A = 2\frac{a+d}{a-d}$ and $B = \frac{4}{a-d}$. Then for any prime p such that $E_{\overline{\mathcal{E}},a,d}$ and $E_{\mathcal{M},A,B}$ have good reduction we have*

$$\#E_{\overline{\mathcal{E}},a,d}(\mathbb{F}_p) = \#E_{\mathcal{M},A,B}(\mathbb{F}_p). \tag{2.14}$$

When selecting good curves, we will need to know that the group structures of $E_{\mathcal{E},a,d}$ and $E_{\mathcal{M},A,B}$ are isomorphic. Since the isomorphism φ preserves the neutral element (it is an isogeny), it is a general fact that it also preserves the group structure. Nevertheless there is an elementary proof as well.

Theorem 2.3.8 (Theorem 7.3,[BL11]). *Fix a field K with $\mathrm{char}(K) \neq 2$. Fix distinct nonzero elements $a, d \in K$. Define $A = 2(a+d)/(a-d)$ and $B = 4/(a-d)$. Define $\varphi : E_{\overline{\mathcal{E}},a,d} \to E_{\mathcal{M},A,B}$ as the bijection in Theorem 2.3.6. Then $\varphi(P_1 + P_2) = \varphi(P_1) + \varphi(P_2)$ for all $P_1, P_2 \in E_{\overline{\mathcal{E}},a,d}(K)$.*

The Small Point Family Bouvier implemented ECM using Montgomery curves on Graphics Processing Units [Z+13]. For this, the computations are sped up if the starting point P_0 has small coordinates. Since the only points $(x : y : z)$ with $x \in \{-1, 0, 1\}$ are the neutral element $(0 : 1 : 0)$, the order 2 element $(0 : 0 : 1)$ and the points $(1, \pm\sqrt{(A \pm 2)/B})$ of order 4, we cannot choose any of them. Hence, we call Small Point family the set of Montgomery curves $E_{\mathcal{M},A,B}$ such that $(2, 1) \in E_{\mathcal{M},A,B}$. This clearly translates into $B = 4A + 10$.

2.3.2 Torsion points

The points of an elliptic curve which have finite order are called torsion points. If the order of a point P divides m we say that P is an m-torsion point. If E/K is an elliptic curve, we write $E[m]$ for the set of m-torsion points and $E(L)[m]$ for the subgroup of L-rational points of $E[m]$, when L is an algebraic extension of K. The structure of $E[m]$ is as follows.

Theorem 2.3.9 ([Sil09],III, Corollary 6.4 (b)). *Let E/K be an elliptic curve and m a non-zero integer. If $\mathrm{char}(K) = 0$ or if m is coprime to $\mathrm{char}(K)$ then we have*

$$E[m] \simeq \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}. \tag{2.15}$$

Suyama [Suy85] proposed a family of Montgomery curves which have a rational point P_3 of order 3 and a rational non-torsion point P_∞, which is used as a starting point for ECM. The basic idea is that if E has a torsion point then, for all but finitely many primes, $E(\mathbb{F}_p)$ has a point of the same order. Heuristically, if the cardinality of a curve is divisible by 3 it has a larger smoothness probability. For now let us prove that the Suyama family imposes that $\#E(\mathbb{F}_p)$ is divisible by 3. For this let us define the reduction of a point.

Definition 2.3.10. Let $E/\mathbb{Q}$ be an elliptic curve and P a rational point on E. Let x_0, y_0 and z_0 be 3 coprime integers such that $P = (x_0 : y_0 : z_0)$. Let p be a prime such that E has good reduction modulo p. We call reduction of P modulo p the point $(\overline{x_0} : \overline{y_0} : \overline{z_0})$ of $E(\mathbb{F}_p)$, where the bar denotes the reduction modulo p.

Since the addition laws of an elliptic curve are given by polynomials, the condition $3 \times P = 0$ can be expressed by a system of polynomial equations, as we explain in detail later. Hence $\overline{P}$ has order 3 or 0. For all but finitely many primes p, the z coordinate of P is coprime to p, so the reduction $\overline{P}$ of P mod p is not the neutral element; it has then order 3.

This result is a particular case of the theorem below. For any elliptic curve $E/\mathbb{Q}$, any number field K and any prime ideal $\mathfrak{p}$ of K we define the reduction modulo $\mathfrak{p}$ of each point $P = (x : y : z) \in E(K)$ as the point of coordinates $(\overline{x} : \overline{y} : \overline{z})$ where the bar represents the residue modulo $\mathfrak{p}$.

Theorem 2.3.11 ([Sil09],VII, Proposition 3.1(b)). *Let E/K be an elliptic curve, $m \geq 1$ an integer and $\mathfrak{p}$ a prime ideal whose norm is coprime to m. Let $k_\mathfrak{p}$ be the residue field of $\mathfrak{p}$. If E has good reduction modulo $\mathfrak{p}$, then the reduction map*

$$\begin{array}{ccc} E(K)[m] & \to & E(k_\mathfrak{p})[m] \\ P & \mapsto & \overline{P} \end{array}$$

is injective.

It is hence natural to search for curves with many rational torsion points. A classical result states that there are no curves $E/\mathbb{Q}$ with more than 16 rational points.

Theorem 2.3.12 (Mazur,[Sil09],VIII, Theorem 7.5)**.** *Let $E/\mathbb{Q}$ be an elliptic curve. Then the torsion subgroup $E_{tors}(\mathbb{Q})$ is one of the following fifteen groups:*

$$\begin{aligned} \mathbb{Z}/N\mathbb{Z} \qquad & 1 \leq N \leq 10 \text{ or } N = 12; \\ \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2N\mathbb{Z} \qquad & 1 \leq N \leq 4. \end{aligned}$$

Parametrization tools A first object that we will use are the division polynomials.

Definition 2.3.13. Let $E_{\mathcal{W},A,B} : y^2 = x^3 + Ax + B$ be an elliptic curve over $\mathbb{Q}$ and $m \geq 2$ an integer. The m-division polynomial P_m is defined as the monic polynomial whose roots are the x-coordinates of all the m-torsion affine points. P_m^{new} is defined as the monic polynomial whose roots are the x-coordinates of the affine points of order exactly m.

Proposition 2.3.14 ([BSS99], Section III.4)**.** *For all $m \geq 2$ we have:*

1. $P_m, P_m^{\text{new}} \in \mathbb{Q}[X]$;
2. $\deg(P_m) = \frac{(m^2+2-3\eta)}{2}$, *where η is the remainder of m modulo 2.*

For example, the 2-division polynomial is $x^3 + Ax + B$. Note that one obtains different division polynomials for other shapes of elliptic curves (short Weierstrass, Montgomery, Edwards, etc.). For example the 2-division polynomial of $E_{\mathcal{M},A,B}$ is $x^3 + Ax^2 + x$. One can rapidly compute any division polynomial using the explicit formulae in Section III.4 of [BSS99].

A second concept used is the genus of a curve, as defined in [Sti08]. Since it can be easily computed using computer algebra software, the genus is a tool to distinguish between three types of curves [SWPD07].

If a plane curve has genus 0 and one has a point (x_0, y_0), then one can find a parametrization expressed by rational fractions $\{(P_x(t), P_y(t)) \mid t \in \mathbb{Q}\}$.

If a curve has genus 1 and one has a point (x_0, y_0), one can put the curve in the form of an elliptic curve. If, in addition, one has a point P of infinite order on the newly obtained elliptic curve, then one can find infinitely many solutions by computing $n \times P$. Although no proven algorithm is known for determining such a point P, Magma implements a relatively fast, non-certified method. Note also that a theorem of Mordell-Weil states that, for any elliptic curve we have

$$E(\mathbb{Q}) \simeq E_{\text{tors}}(\mathbb{Q}) \times \mathbb{Z}^r \tag{2.16}$$

for a finite number r. This latter value is called the rank of the elliptic curve E. Hence, an elliptic curve has an infinity of rational points if and only if its rank is positive.

Finally, if an equation cannot be transformed into a conic or an elliptic curve, then it has a finite number of rational points.

Theorem 2.3.15 (Faltings). *Any curve of genus $g \geq 2$ has a finite number of rational points.*

In short, if one searches for infinitely many solutions of an equation of 2 unknowns one can start by computing the genus of the equation. If it is 0, one obtains a rational parametrization. If it is 1 we have an elliptic curve and, if additionally the rank is positive, one obtains a parametrization indexed by $\mathbb{Z}$. Finally, if the genus is 2 or larger, then one concludes that there are only finitely many solutions.

Suyama family The end of this chapter partly reproduces [?], which is a joint work with Bouvier, Bos, Kleinjung and Montgomery. Let us see how to obtain the parametrization of Suyama. For this let us put $E(A, B, x, y) = By^2 - (x^3 + Ax^2 + x)$. We want to find A and B such that there exists a point (x_3, y_3) on the Montgomery curve $E_{\mathcal{M},A,B}$. By the definition of the division polynomial P_3, this translates into $P_3(x_3) = 0$ and $E(A, B, x_3, y_3) = 0$. Next, we want a second point (x_∞, y_∞) on $E_{\mathcal{M},A,B}$ so we impose $E(A, B, x_\infty, y_\infty) = 0$. This counts for 6 unknowns, A, B, x_3, y_3, x_∞ and y_∞, and 4 equations. The solutions describe a surface on which we can search for as many solutions as needed, but we do not have a method to parametrize the set of solutions. Suyama imposed a new ad-hoc equation so that the polynomial system reduces to one equation with two unknowns. The further computations are facilitated by adding an extra unknown k with the extra equation $k = y_3/y_\infty$.

Definition 2.3.16. A Montgomery curve $E_{\mathcal{M},A,B}$ for which there exist $x_3, y_3, k, x_\infty, y_\infty \in \mathbb{Q}$ such that

$$\begin{cases} P_3(x_3) = 0, \quad By_3^2 = x_3^3 + Ax_3^2 + x_3 & \text{(3-torsion point)} \\ k = \dfrac{y_3}{y_\infty}, \quad k^2 = \dfrac{x_3^3 + Ax_3^2 + x_3}{x_\infty^3 + Ax_\infty^2 + x_\infty} & \text{(non-torsion point)} \\ x_\infty = x_3^3. & \text{(Suyama equation)} \end{cases} \tag{2.17}$$

is called a Suyama curve.

Using the expression of the order-3 division polynomial, $P_3(x) = x^4 + \frac{4}{3}Ax^3 + 2x^2 - \frac{1}{3}$, we obtain A as a rational fraction $A(x_3)$ of x_3. Then k and x_3 must satisfy the equation

$$C(x_3, k) : (x_3^6 + A(x_3)x_3^3 + 1) - k^2(x_3^2 + A(x_3)x_3 + 1) = 0. \tag{2.18}$$

If we factor C we obtain

$$C = (x_3 - 1)(x_3 + 1)\left(x_3^4 + \frac{5}{4}x_3^2 - \frac{1}{4}k^2\right). \tag{2.19}$$

The values $x_3 = 1$ and $x_3 = -1$ are forbidden since they correspond to 4-torsion points. The last factor of C has genus 0 and it leads to a rational parametrization. For any rational parameter $\sigma \in \mathbb{Q} \setminus \left\{0, \pm 1, \pm 3, \pm 5, \pm\frac{5}{3}\right\}$, we put:

$$\begin{cases} x_3 &= -\frac{\sigma}{4}(\frac{5}{\sigma^2} - 1) \\ k &= \frac{\sigma^4 - 25}{8\sigma^2}. \end{cases} \tag{2.20}$$

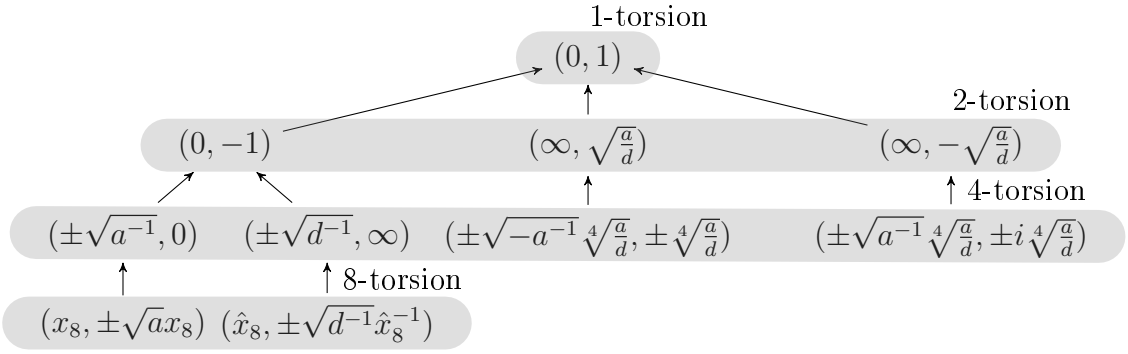


Figure 2.2: An overview of all 1-, 2- and 4-torsion and some 8-torsion points on twisted Edwards curves. x_8 and $\hat{x}_8$ in the 8-torsion points are such that $adx_8^4 - 2ax_8^2 + 1 = 0$ and $ad\hat{x}_8^4 - 2d\hat{x}_8^2 + 1 = 0$.

Finally, using Equation 2.17, we obtain the values of A originally given by Suyama [Suy85], [ZD06]:

$$u = \sigma^2 - 5, \quad v = 4\sigma, \quad A = -2 - (3u+v)(u-v)^3/(4vu^3), \quad x_\infty = u^3/v^3. \tag{2.21}$$

We omit the formula for B as it is not used when computing on Montgomery curves.

Montgomery/twisted Edwards curves with large torsion groups in $\mathbb{Q}$. Suyama noted that any Montgomery curve E has a rational 2-torsion point, whereas, for any prime p, $E(\mathbb{F}_p)$ has a subgroup $\mathbb{Z}/4\mathbb{Z}$ or a subgroup $\mathbb{Z}/2\mathbb{Z}\times\mathbb{Z}/2\mathbb{Z}$. Due to the Montgomery-twisted Edwards equivalence, the same property is shared by twisted Edwards curves. In order to understand this result one can compute the 4-torsion points of $E_{\overline{\mathcal{E}},a,d}$ in the algebraic closure of $\mathbb{Q}(a,d)$, i.e., the fraction field of two variables.

Theorem 2.3.17 ([BBLP13]). *Let $\mathbb{Q}(a,d)$ be the field of rational fractions in two variables. If x_8 and $\hat{x}_8$ are such that $adx_8^4 - 2ax_8^2 + 1 = 0$ and $ad\hat{x}_8^4 - 2d\hat{x}_8 + 1 = 0$. Then the complete list of 4-torsion points and a partial list of 8-torsion list is as in Figure 2.2.*

Hence, if a/d is a square, $E_{\overline{\mathcal{E}},a,d}$ has four 2-torsion points. If a or d is a square, $E_{\overline{\mathcal{E}},a,d}$ has at least one point of order 4. Since a, d and a/d multiply to a square, they cannot be simultaneously non-squares modulo a prime. Hence, any twisted Edwards curve, reduced modulo any prime of good reduction p has at least 4 points of 4-torsion. If one considers the cases when some of the expressions are squares, one obtains a stronger result.

Theorem 2.3.18. *Let $E = E_{\overline{\mathcal{E}},a,d}$ be a completed twisted Edwards curve (resp. a Montgomery curve $E_{\mathcal{M},A,B}$) over $\mathbb{Q}$. Let p be a prime such that E has good reduction at p.*

1. *If* $p \equiv 3 \pmod 4$ *and* $\frac{a}{d}$ *(resp.* $A^2 - 4$*) is a quadratic residue modulo* p*, then* $E(\mathbb{F}_p)[4] \simeq \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$*;*

2. *If* $p \equiv 1 \pmod 4$*,* a *(resp.* $\frac{A+2}{B}$*) is a quadratic residue modulo* p *(in particular* $a = \pm 1$*) and* $\frac{a}{d}$ *(resp.* $A^2 - 4$*) is a quadratic residue modulo* p*, then* $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z} \subset E(\mathbb{F}_p)[4]$*;*

3. *If* $p \equiv 1 \pmod 4$*,* $\frac{a}{d}$ *(resp.* $A^2 - 4$*) is a quadratic non-residue modulo* p *and* $a - d$ *(resp.* B*) is a quadratic residue modulo* p*, then* $E(\mathbb{F}_p)[8] \simeq \mathbb{Z}/8\mathbb{Z}$*.*

Proof. Use the results illustrated in Figure 2.2. □

Montgomery obtained an elliptic parametrizations for the families of curves with a non-torsion rational point and with a rational torsion group of $\mathbb{Z}/12\mathbb{Z}$ and $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$ respectively [Mon92]. Moreover, he gave experimental evidence that the family of 12 rational points has better results than the Suyama family. Bernstein et al. [BBJ+08] translated these families into the language of twisted Edwards curves. Finally, Bernstein and Lange [BBL10] presented a family of rational torsion group $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ when $a = -1$. They also gave experimental evidence that the translation of the Suyama family to the case of $a = -1$ twisted Edwards curves has equal performances as the, expected better, family of 12 rational points. Understanding this surprising behaviour of the $a = -1$ twisted Edwards curves is one of the motivations for the following chapter.

Chapter 3

Finding ECM-friendly curves through a study of Galois properties

We have seen in the previous chapter that ECM is faster when one uses elliptic curves with many torsion points over $\mathbb{Q}$. This chapter, which is a joint work with Cyril Bouvier, Joppe Bos, Thorsten Kleinjung and Peter Montgomery [?], goes beyond the heuristic arguments used before in the literature. We establish a link between the good properties of an elliptic curve and the Galois group of its division polynomials. In particular we explain some good properties observed in practice. It further allows us to improve ECM by proposing new families of elliptic curves. Among other, it accelerates the computations of the smoothing step studied in Chapter 4.
We organize the chapter as follows. We start by recalling our main tool: Chebotarev's theorem. It gives us the probability that an elliptic curve has a given torsion group when reduced modulo a random prime. Of course one still has to compute the explicit Galois group of the division polynomials. We then use this machinery to provide new families of ECM-friendly curves.

3.1 Galois properties of torsion points of elliptic curves

In this section we give a systematic way to compute the probability that the order of a given elliptic curve reduced by an arbitrary prime is divisible by a certain prime power.

3.1.1 Torsion properties of elliptic curves.

Definition 3.1.1. Let K be a finite Galois extension of $\mathbb{Q}$, p a prime and $\mathfrak{p}$ a prime ideal above p with residue field $k_\mathfrak{p}$. The decomposition group $\mathrm{Dec}(\mathfrak{p})$ of $\mathfrak{p}$ is the subgroup of $\mathrm{Gal}(K/\mathbb{Q})$ which stabilizes $\mathfrak{p}$. Call $\alpha^{(\mathfrak{p})}$ the canonical morphism from $\mathrm{Dec}(\mathfrak{p})$ to $\mathrm{Gal}(k_\mathfrak{p}/\mathbb{F}_p)$ and let $\phi_\mathfrak{p}$ be the Frobenius automorphism on the field $k_\mathfrak{p}$. We define $\mathrm{Frobenius}(p) = \bigcup_{\mathfrak{p}|p}(\alpha^{(\mathfrak{p})})^{-1}(\phi_\mathfrak{p})$.

In order to state Chebotarev's theorem we say that a set S of primes admits a natural density equal to δ and we write $\text{Prob}(S) = \delta$ if $\lim_{N\to\infty} \frac{\#(S \bigcap \Pi(N))}{\#\Pi(N)}$ exists and equals δ, where $\Pi(N)$ is the set of primes up to N. If event(p) is a property which can be defined for all primes except a finite set (thus of zero density), when we note Prob(event(p)) we tacitly exclude the primes where event(p) cannot be defined.

Theorem 3.1.2 (Chebotarev, [Neu86]). *Let K be a finite Galois extension of $\mathbb{Q}$. Let $H \subset \text{Gal}(K/\mathbb{Q})$ be a conjugacy class. Then*

$$\text{Prob}\left(\text{Frobenius}(p) = H\right) = \frac{\#H}{\#\,\text{Gal}(K/\mathbb{Q})}.$$

We illustrate the theorem by proving a classical result.

Corollary 3.1.3 (Frobenius' theorem). *Let f be a monic polynomial in $\mathbb{Z}[x]$, not-necessarily irreducible. Let G be the Galois group of f, i.e., the Galois group of the splitting field of f. We call cycle pattern of a permutation σ the list of cardinalities of the orbits of σ. We also call factorization pattern of a polynomial the list of degrees of its irreducible factors. Then the natural density of the primes p such that $f \bmod p$ has a certain factorization pattern $(n_1, n_2, \ldots)$ equals the proportion of elements in G which permute the roots of f with a cycle pattern $(n_1, n_2, \ldots)$.*

For example, the polynomial $x^2 - 2$ splits when reduced modulo half of the primes. An example from [SLJ96] is $f = x^4 + 3x^2 + 7x + 4$. The Galois group of f has 12 elements, which act as permutations on the roots of f with cycle patterns as follows: $(1, 3)$ for $2/3$ of the elements, $(2, 2)$ for $1/4$ of the elements and $(1, 1, 1, 1)$ for the identity element. The theorem states the proportion of primes, for which the modular reduction of f has a factorization pattern $(1, 3)$, $(2, 2)$ or $(1, 1, 1, 1)$. In particular f is reducible for all primes except for a set of zero density. The proof of Corollary 3.1.3 is aimed to illustrate Frobenius(p).

Proof. Call K the splitting field of f and let T be a defining polynomial for K and θ a root of T in K. We choose polynomials $\alpha_i(x)$ with $i \in [1, \deg f]$ such that the roots of f in K are $\alpha_i(\theta)$. Since T is a Galois polynomial, the automorphisms of K are given by $\theta \mapsto \sigma(\theta)$ where σ is a polynomial in $\mathbb{Q}[x]$ such that $T(\sigma(x)) \equiv 0 \bmod T(x)$.

Let p be a prime which does not divide the discriminant $\text{Disc}(T)$. Let $T \equiv \prod_i T_i \bmod p$ be the factorization of $T \bmod p$, which has factors of equal degrees because T is a Galois polynomial. Since p does not divide $\text{Disc}(T)$, the prime ideals of K over p are $\mathfrak{p}_i := \langle p, T_i(\theta)\rangle$ with T_i divisors of $T \bmod p$.

Let ϕ_p be the Frobenius map in $\overline{F_p}$. We represent the field F of $p^{\deg T_1}$ elements as $\mathbb{F}_p[y]/\langle T_1(y)\rangle$. We call $\overline{\theta}$ the element $y \bmod \langle p, T_1\rangle$, which is a root of T_1 in F. Note that the roots of T in F are $\sigma(\overline{\theta})$ with $\sigma \in \text{Gal}(K)$. We call Frobenius of $\mathfrak{p}_1$ the automorphism $x \mapsto \sigma(x)$ of K such that $\sigma(\overline{\theta}) = \phi_p(\overline{\theta})$. Frobenius$(p)$ has as many elements as factors of $T \bmod p$.

Note that the roots of f in F are $\{\alpha_i(\overline{\theta}) \mid \alpha_i(\theta) \text{ is root of } f \text{ in } K\}$. Let us show that the factorization pattern of $f \bmod p$ coincides with the cycle pattern of the Frobenius of $\mathfrak{p}_1$ on the roots of f.

Indeed, each irreducible factor of f mod p corresponds to an orbit of ϕ_p on the roots of f. By the choice of σ, ϕ_p coincides with $\overline{\theta} \mapsto \sigma(\overline{\theta})$ on the roots of f. Hence ϕ_p has the same orbits on the roots $\alpha_i(\overline{\theta})$ of f in F as $\theta \mapsto \sigma(\theta)$ does on the roots of f in K.

We conclude by applying Chebotarev's theorem. The proportion of primes p whose Frobenius has a given cycle pattern on the roots of f in F is the proportion of elements of G which have the given cycle pattern on the roots of f in K. □

Before applying Chebotarev's theorem to the case of elliptic curves, we introduce a notation. For every elliptic curve E over a field F and all $m \in \mathbb{N}$, $m \geq 2$, we consider the field $F(E[m])$ which is the smallest extension of F containing all the m-torsion of E. The next result is classical, but we present its proof for the intuition it brings.

Proposition 3.1.4. *For every integer $m \geq 2$ and any elliptic curve E over some field F, the following hold:*

1. *$F(E[m])/F$ is a Galois extension;*
2. *there is an injective morphism $\iota_m : \mathrm{Gal}(F(E[m])/F) \hookrightarrow \mathrm{Aut}(E(\overline{F})[m])$.*

Proof. (1) Since the addition law of E can be expressed by rational functions over F, there exist polynomials $f_m, g_m \in F[X,Y]$ such that the coordinates of the points in $E(\overline{F})[m]$ are the solutions of the system $(f_m = 0, g_m = 0)$. Therefore $F(E[m])$ is the splitting field of $\mathrm{Res}_X(f_m, g_m)$ and $\mathrm{Res}_Y(f_m, g_m)$ and in particular is Galois.
(2) For each $\sigma \in \mathrm{Gal}(F(E[m])/F)$ we call $\iota_m(\sigma)$ the application which sends $(x,y) \in E(\overline{F})[m]$ into $(\sigma(x), \sigma(y))$. Thanks to the discussion above, $\iota_m(\sigma)$ sends the points of $E(\overline{F})[m]$ in $E(\overline{F})[m]$. Since the addition law can be expressed by rational functions over F, for each σ, $\iota_m(\sigma) \in \mathrm{Aut}(E(\overline{F})[m])$. One easily checks that the kernel of ι_m is the identity. □

Notation 3.1.5. We choose two points which generate $E(\overline{\mathbb{Q}})[m]$ (see Theorem 2.3.9), thereby inducing an isomorphism

$$\psi_m : \mathrm{Aut}(E(\overline{\mathbb{Q}})[m]) \to \mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z}).$$

Let ι_m be the injection given by Proposition 3.1.4. We call

$$\rho_m : \mathrm{Gal}(\mathbb{Q}(E[m])/\mathbb{Q}) \to \mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z})$$

the injective morphism $\psi_m \circ \iota_m$.

Remark 3.1.6. Note that $\#\mathrm{Gal}(\mathbb{Q}(E[m])/\mathbb{Q})$ is bounded by $\#\mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z})$. For every prime π, we have $\#\mathrm{GL}_2(\mathbb{Z}/\pi\mathbb{Z}) = (\pi-1)^2(\pi+1)\pi$, and for every integer $k \geq 1$, $\#\mathrm{GL}_2(\mathbb{Z}/\pi^{k+1}\mathbb{Z}) = \pi^4 \#\mathrm{GL}_2(\mathbb{Z}/\pi^k\mathbb{Z})$.

Let p be a prime such that E has good reduction at p and $p \nmid m$. Let $\iota_m^{(p)}$ be the injection of $\mathrm{Gal}(\mathbb{F}_p(E[m])/\mathbb{F}_p)$ into $\mathrm{Aut}(E(\overline{\mathbb{F}_p})[m])$ given by Proposition 3.1.4. Theorem 2.3.11 applied to $K = \mathbb{Q}(E[m])$ implies the existence of a canonical isomorphism from $\mathrm{Aut}(E(\overline{\mathbb{Q}})[m])$ to $\mathrm{Aut}(E(\overline{\mathbb{F}_p})[m])$ for each prime ideal $\mathfrak{p}$ above p.

Notation 3.1.7. For all $g \in \mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z})$ we put $\mathrm{Fix}(g) = \{v \in (\mathbb{Z}/m\mathbb{Z})^2 \mid g(v) = v\}$. Note that any element conjugated to g has an isomorphic group of fixed elements. If we are interested only in the isomorphism class we use the notation $\mathrm{Fix}(C)$ where C is a set of conjugated elements. We use analogous notations for $\mathrm{Aut}(E(\overline{\mathbb{Q}})[m])$ and $\mathrm{Aut}(E(\overline{\mathbb{F}_p})[m])$.

Theorem 3.1.8. *Let E be an elliptic curve over $\mathbb{Q}$ and $m \geq 2$ be an integer. Put $K = \mathbb{Q}(E[m])$. Let T be a subgroup of $\mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}$. Then,*

1. $$\mathrm{Prob}(E(\mathbb{F}_p)[m] \simeq T) = \frac{\#\{g \in \rho_m(\mathrm{Gal}(K/\mathbb{Q})) \mid \mathrm{Fix}(g) \simeq T\}}{\#\,\mathrm{Gal}(K/\mathbb{Q})}.$$

2. *Let $a, n \in \mathbb{N}$ such that $a \leq n$ and $\gcd(a, n) = 1$ and let ζ_n be a primitive nth root of unity. Put $G_a = \{\sigma \in \mathrm{Gal}(K(\zeta_n)/\mathbb{Q}) \mid \sigma(\zeta_n) = \zeta_n^a\}$. Then:*
$$\mathrm{Prob}(E(\mathbb{F}_p)[m] \simeq T \mid p \equiv a \bmod n) = \frac{\#\{\sigma \in G_a \mid \mathrm{Fix}(\rho_m(\sigma_{|K})) \simeq T\}}{\#G_a}.$$

Proof. (1) Let $p \nmid m$ be a prime for which E has good reduction and let $\mathfrak{p}$ be a prime ideal of K over p. We abbreviate $H = \{\sigma \in \mathrm{Gal}(K/\mathbb{Q}) \mid \mathrm{Fix}(\iota_m(\sigma)) \simeq T\}$. First note that $E(\mathbb{F}_p)[m] = \mathrm{Fix}(\iota_m^{(p)}(\phi_p))$ where ϕ_p is the Frobenius in $\mathrm{Gal}(\mathbb{F}_p(E[m])/\mathbb{F}_p)$. Since the diagram

$$\begin{array}{ccccc} \mathrm{Dec}(\mathfrak{p}) & \hookrightarrow & \mathrm{Gal}(\mathbb{Q}(E[m])/\mathbb{Q}) & \xhookrightarrow{\iota_m} & \mathrm{Aut}(E(\overline{\mathbb{Q}})[m]) \\ \downarrow{\scriptstyle \alpha^{(\mathfrak{p})}} & & & & \downarrow \\ \mathrm{Gal}(k_{\mathfrak{p}}/\mathbb{F}_p) & \xrightarrow{\sim} & \mathrm{Gal}(\mathbb{F}_p(E[m])/\mathbb{F}_p) & \xhookrightarrow{\iota_m^{(p)}} & \mathrm{Aut}(E(\overline{\mathbb{F}_p})[m]) \end{array}$$

is commutative and since $\mathrm{Frobenius}(p) \subset \mathrm{Gal}(K/\mathbb{Q})$ is the conjugacy class generated by $(\alpha^{(\mathfrak{p})})^{-1}(\phi_{\mathfrak{p}})$ we have $E(\mathbb{F}_p)[m] \simeq \mathrm{Fix}(\iota_m(\mathrm{Frobenius}(p)))$.

Decompose H into a disjoint union of conjugacy classes $C_1, \ldots, C_N$. Then $\mathrm{Fix}(\iota_m(\mathrm{Frobenius}(p))) \simeq T$ is equivalent to $\mathrm{Frobenius}(p)$ being one of the C_i. Thanks to Theorem 3.1.2 we obtain:

$$\mathrm{Prob}(E(\mathbb{F}_p)[m] \simeq T) = \sum_{i=1}^{N} \mathrm{Prob}(\mathrm{Frobenius}(p) = C_i) = \sum_{i=1}^{N} \frac{\#C_i}{\#\,\mathrm{Gal}(K/\mathbb{Q})} = \frac{\#H}{\#\,\mathrm{Gal}(K/\mathbb{Q})}.$$

(2) Using similar arguments as in (1) we have to evaluate

$$\frac{\mathrm{Prob}(\mathrm{Frobenius}(p) \in \{C_1, \ldots, C_N\}, p \equiv a \bmod n)}{\mathrm{Prob}(p \equiv a \bmod n)}.$$

Let p be a prime and $\mathfrak{p}$ a prime ideal as in the first part of the proof, and let $\mathfrak{P}$ be a prime ideal of $K(\zeta_n)$ lying over $\mathfrak{p}$. Furthermore let $\tilde{C}_1, \ldots, \tilde{C}_{\tilde{N}}$ be the conjugacy classes of $\mathrm{Gal}(K(\zeta_n)/\mathbb{Q})$ that are in the pre-images of $C_1, \ldots, C_N$ and whose elements σ satisfy $\sigma(\zeta_n) = \zeta_n^a$. Since $\mathrm{Gal}(K(\zeta_n)/\mathbb{Q})$ maps ζ_n to primitive nth roots of unity we have for $\sigma \in (\alpha^{(\mathfrak{P})})^{-1}(\phi_{\mathfrak{P}})$ that $\sigma(\zeta_n) = \zeta_n^b$ holds for an integer b. Together with $\sigma(x) \equiv x^p \bmod \mathfrak{P}$ we get $\zeta_n^b \equiv \zeta_n^p \bmod \mathfrak{P}$. If we exclude the finitely many primes dividing the norms of $\zeta_n^c - 1$ for $c = 1, \ldots, n-1$ we

obtain $b \equiv p \bmod n$. Since Frobenius$(K(\zeta_n), p)$, the Frobenius conjugacy class for $K(\zeta_n)$, is the pre-image of Frobenius(p), we get with the argument above Prob(Frobenius$(p) \in \{C_1, \ldots, C_N\}, p \equiv a \bmod n) =$ Prob(Frobenius$(K(\zeta_n), p) \in \{\tilde{C}_1, \ldots, \tilde{C}_{\tilde{N}}\}$). A similar consideration for the denominator Prob$(p \equiv a \bmod n)$ completes the proof. □

Remark 3.1.9. Put $K = \mathbb{Q}(E[m])$. If $[K(\zeta_n) : \mathbb{Q}(\zeta_n)] = [K : \mathbb{Q}]$, then one has Prob$(E(\mathbb{F}_p)[m] \simeq T \mid p \equiv a \bmod n) =$ Prob$(E(\mathbb{F}_p)[m] \simeq T)$ for a coprime to n. Indeed, according to Galois theory, $\mathrm{Gal}(K(\zeta_n)/\mathbb{Q})/\mathrm{Gal}(K(\zeta_n)/K) \simeq \mathrm{Gal}(K/\mathbb{Q})$ through $\overline{\sigma} \mapsto \sigma_{|K}$. Since $[K(\zeta_n) : \mathbb{Q}(\zeta_n)] = [K : \mathbb{Q}]$, we have $[K(\zeta_n) : K] = \varphi(n)$ and therefore each element σ of $\mathrm{Gal}(K/\mathbb{Q})$ extends in exactly one way to an element of $\mathrm{Gal}(K(\zeta_n)/\mathbb{Q})$ which satisfies $\sigma(\zeta_n) = \zeta_n^a$. Note that for $n \in \{3, 4\}$ the condition is equivalent to $\zeta_n \notin K$.

The families constructed by Brier and Clavier [BC10], which are dedicated to integers N such that the nth cyclotomic polynomial has roots modulo all prime factors of N, modify $[K(\zeta_n) : \mathbb{Q}(\zeta_n)]$ by imposing a large torsion subgroup over $\mathbb{Q}(\zeta_n)$.

An important particular case of Theorem 3.1.8 is as follows:

Corollary 3.1.10. *Let E be an elliptic curve and π be a prime number. Then,*

$$\mathrm{Prob}\left(E(\mathbb{F}_p)[\pi] \simeq \mathbb{Z}/\pi\mathbb{Z}\right) = \frac{\#\{g \in \rho_\pi(\mathrm{Gal}(\mathbb{Q}(E[\pi])/\mathbb{Q})) \mid \det(g - \mathrm{Id}) = 0, g \neq \mathrm{Id}\}}{\#\,\mathrm{Gal}(\mathbb{Q}(E[\pi])/\mathbb{Q})},$$

$$\mathrm{Prob}\left(E(\mathbb{F}_p)[\pi] \simeq \mathbb{Z}/\pi\mathbb{Z} \times \mathbb{Z}/\pi\mathbb{Z}\right) = \frac{1}{\#\,\mathrm{Gal}(\mathbb{Q}(E[\pi])/\mathbb{Q})}.$$

Example 3.1.11. Let us compute these probabilities for the curves $E_1 : y^2 = x^3 + 5x + 7$ and $E_2 : y^2 = x^3 - 11x + 14$ and the primes $\pi = 3$ and $\pi = 5$. Here E_1 illustrates the generic case, whereas E_2 has special Galois groups. One checks with Sage [S+11] that $[\mathbb{Q}(E_1[3]) : \mathbb{Q}] = 48$ and $\#\,\mathrm{GL}_2(\mathbb{Z}/3\mathbb{Z}) = 48$. By Proposition 3.1.4 we deduce that $\rho_3(\mathrm{Gal}(\mathbb{Q}(E_1[3])/\mathbb{Q})) = \mathrm{GL}_2(\mathbb{Z}/3\mathbb{Z})$. A simple computation shows that $\mathrm{GL}_2(\mathbb{Z}/3\mathbb{Z})$ contains 21 elements having 1 as eigenvalue, one of which is Id. Corollary 3.1.10 gives the following probabilities: Prob$(E_1(\mathbb{F}_p)[3] \simeq \mathbb{Z}/3\mathbb{Z}) = \frac{20}{48}$ and Prob$(E_1(\mathbb{F}_p)[3] \simeq \mathbb{Z}/3\mathbb{Z} \times \mathbb{Z}/3\mathbb{Z}) = \frac{1}{48}$. We used the same method for all the probabilities of Table 3.1, where we compare them to experimental values.

Note that the relative difference between theoretical and experimental values never exceeds 0.4%. It is interesting to observe that reducing the Galois group does not necessarily increase the probabilities, as it is shown for $\pi = 3$.

3.1.2 Effective computations of $\mathbb{Q}(E[m])$ and $\rho_m(\mathrm{Gal}(\mathbb{Q}(E[m])/\mathbb{Q}))$ for prime powers.

Recall the definition of the division polynomials given in Definition 2.3.13. One can compute $\mathbb{Q}(E[\pi])$ for any prime $\pi \geq 3$ using the following method:

		E_1	E_2
$\#\,\mathrm{GL}_2(\mathbb{Z}/3\mathbb{Z})$		48	
$\#\,\mathrm{Gal}(\mathbb{Q}(E[3])/\mathbb{Q})$		48	16
$\mathrm{Prob}(E(\mathbb{F}_p)[3] \simeq \mathbb{Z}/3\mathbb{Z} \times \mathbb{Z}/3\mathbb{Z})$	Th.	$\frac{1}{48} \approx 0.02083$	$\frac{1}{16} = 0.06250$
	Exp.	0.02082	0.06245
$\mathrm{Prob}(E(\mathbb{F}_p)[3] \simeq \mathbb{Z}/3\mathbb{Z})$	Th.	$\frac{20}{48} \approx 0.4167$	$\frac{4}{16} = 0.2500$
	Exp	0.4165	0.2501
$\#\,\mathrm{GL}_2(\mathbb{Z}/5\mathbb{Z})$		480	
$\#\,\mathrm{Gal}(\mathbb{Q}(E[5])/\mathbb{Q})$		480	32
$\mathrm{Prob}(E(\mathbb{F}_p)[5] \simeq \mathbb{Z}/5\mathbb{Z} \times \mathbb{Z}/5\mathbb{Z})$	Th.	$\frac{1}{480} \approx 0.002083$	$\frac{1}{32} = 0.03125$
	Exp.	0.002091	0.03123
$\mathrm{Prob}(E(\mathbb{F}_p)[5] \simeq \mathbb{Z}/5\mathbb{Z})$	Th.	$\frac{114}{480} = 0.2375$	$\frac{10}{32} = 0.3125$
	Exp.	0.2373	0.3125

Table 3.1: Comparison of the theoretical values (Th) of Corollary 3.1.10 to the experimental results for all primes below 2^{25} (Exp).

1. Make a first extension of $\mathbb{Q}$ through an irreducible factor of P_π and obtain a number field F_1 where P_π has a root α_1.

2. Let $f_2(y) = y^2 - (\alpha_1^3 + A\alpha_1 + B) \in F_1[y]$ and F_2 be the extension of F_1 by f_2. F_2 contains a π-torsion point M_1. In F_2, P_π has $\frac{\pi-1}{2}$ trivial roots representing the x coordinates of the multiples of M_1.

3. Call F_3 the extension of F_2 through an irreducible factor of $P_\pi \in F_2[x]$ other than those corresponding to the trivial roots.

4. Let α_2 be the new root of P_π in F_3. Let $f_4(y) = y^2 - (\alpha_2^3 + A\alpha_2 + B) \in F_3[y]$ and F_4 be the extension of F_3 through f_4. F_4 contains all the π-torsion.

The case of prime powers π^k with $k \geq 2$ is handled recursively. Having computed $\mathbb{Q}(E[\pi^{k-1}])$, we obtain $\mathbb{Q}(E[\pi^k])$ by repeating the 4 steps above with $P_{\pi^k}^{\text{new}}$ instead of P_π and by considering as trivial roots all the x-coordinates of the points $\{P + M_1 \mid P \in E[\pi^{k-1}]\}$.

In practice, we observed that in general $P_\pi, f_2, P_\pi^{(F_2)}$ and f_4 are irreducible, where $P_\pi^{(F_2)}$ is P_π divided by the factors corresponding to the trivial roots. If this is the case, as $\deg(P_\pi) = \frac{\pi^2-1}{2}$ (Proposition 2.3.14), the absolute degree of F_4 is $\frac{\pi^2-1}{2} \cdot 2 \cdot \frac{\pi^2-\pi}{2} \cdot 2 = (\pi-1)^2(\pi+1)\pi$. By Remark 3.1.6, $\#\,\mathrm{GL}_2(\mathbb{Z}/\pi\mathbb{Z}) = (\pi-1)^2(\pi+1)\pi$, therefore in general we expect $\rho_\pi(\mathrm{Gal}(\mathbb{Q}(E[\pi])/\mathbb{Q})) = \mathrm{GL}_2(\mathbb{Z}/\pi\mathbb{Z})$. Also, we observed that in general the degree of the extension $\mathbb{Q}(E[\pi^k])/\mathbb{Q}(E[\pi^{k-1}])$ is π^4.

Serre [Ser71] proved that the observations above are almost always true. The next theorem is a restatement of items (1) and (6) in the introduction of [Ser71].

Theorem 3.1.12 (Serre). *Let E be an elliptic curve without complex multiplication.*

1. *For all primes π and $k \geq 1$ the index $[\mathrm{GL}_2(\mathbb{Z}/\pi^k\mathbb{Z}) : \rho_{\pi^k}(\mathrm{Gal}(\mathbb{Q}(E[\pi^k])/\mathbb{Q}))]$ is non-decreasing and bounded by a constant depending on E and π.*

2. *For all primes π outside a finite set depending on E and for all $k \geq 1$, $\rho_{\pi^k}(\mathrm{Gal}(\mathbb{Q}(E[\pi^k])/\mathbb{Q}) = \mathrm{GL}_2(\mathbb{Z}/\pi^k\mathbb{Z})$.*

Definition 3.1.13. Put $I(E,\pi,k) = [\mathrm{GL}_2(\mathbb{Z}/\pi^k\mathbb{Z}) : \rho_{\pi^k}(\mathrm{Gal}(\mathbb{Q}(E[\pi^k])/\mathbb{Q}))]$. If E does not admit complex multiplication, we call Serre's exponent the integer $n(E,\pi) = \min\{n \in \mathbb{N}^* \mid \forall k \geq n, I(E,\pi,k+1) = I(E,\pi,k)\}$.

In [Ser81] Serre showed that in some cases one can prove that $I(E,\pi,k) = 1$ for all positive integers k. Indeed, Serre proved that the surjectivity of ρ_{π^k} (or equivalently $I(E,\pi,k) = 1$) follows from the surjectivity of ρ_π (or equivalently $I(E,\pi,1) = 1$) for all rational elliptic curves E without complex multiplication and for all primes $\pi \geq 5$. In order to have the same kind of results for $\pi = 2$ (resp. $\pi = 3$) one has to suppose that ρ_2, ρ_4 and ρ_8 are surjective (resp. ρ_3 and ρ_9 are surjective).

Serre also conjectured that only a finite number of primes, not depending on the curve E, can occur in the second point of Theorem 3.1.12. The current conjecture is that for all rational elliptic curves without complex multiplication and all primes $\pi \geq 37$, ρ_π is surjective. In [Zyw11] Zywina described an algorithm that computes the primes π for which ρ_π is not surjective and checked the conjecture for all elliptic curves in Magma's database (currently this covers curves with conductor at most 14000). [1]

Remark 3.1.14. One application of Serre's results is as follows. Experiments show that an elliptic curve without complex multiplication is close to a cyclic group when reduced modulo an arbitrary prime, regardless on its rank over $\mathbb{Q}$. For a given bound B, computing $\mathrm{Prob}(\exists \pi > B, \mathbb{Z}/\pi\mathbb{Z} \times \mathbb{Z}/\pi\mathbb{Z} \subset E(\mathbb{F}_p))$ goes beyond the scope of this paper. Still, we can compute $\mathrm{Prob}(\mathbb{Z}/\pi\mathbb{Z} \times \mathbb{Z}/\pi\mathbb{Z} \subset E(\mathbb{F}_p))$ for every π such that ρ_π is surjective. Indeed, for these primes, Corollary 3.1.10 gives the probability $\mathrm{Prob}(\mathbb{Z}/\pi \times \mathbb{Z}/\pi \subset E(\mathbb{F}_p)) = \frac{1}{\pi(\pi+1)(\pi-1)^2}$.

The method described above allows us to compute $\mathbb{Q}(E[m])$ as an extension tower. Then it is easy to obtain its absolute degree and a primitive element. Identifying $\rho_\pi(\mathrm{Gal}(\mathbb{Q}(E[m])/\mathbb{Q}))$ (up to conjugacy) is easy when there is only one subgroup (up to conjugacy) of $\mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z})$ with the right order. In the other case we check for each $g \in \mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z})$ using the fixed generators of $E(\overline{\mathbb{Q}})[m]$ whether g gives rise to an automorphism on $\mathbb{Q}(E[m])$. In practice, the bottleneck of this method is the factorization of polynomials with coefficients over number fields.

A faster probabilistic algorithm for computing $\mathrm{Gal}(\mathbb{Q}(E[\pi])/\mathbb{Q})$ was proposed by Sutherland [Sut12].

3.1.3 Divisibility by a prime power

It is a common fact that, for a given prime π, the cardinality of an arbitrary elliptic curve over $\mathbb{F}_p$ has a slightly larger probability to be divisible by π than an arbitrary integer of size p. In this subsection we will rigorously compute those probabilities under some hypothesis of generality.

[1] We thank Andrew Sutherland for bringing this article to our attention.

Notation 3.1.15. Let π be a prime and $i, j, k \in \mathbb{N}$ such that $i \leq j$. We put:

$$p_{\pi,k}(i,j) = \text{Prob}(E(\mathbb{F}_p)[\pi^k] \simeq \mathbb{Z}/\pi^i\mathbb{Z} \times \mathbb{Z}/\pi^j\mathbb{Z}).$$

Let $\ell \leq m$ be integers. When it is defined we denote:

$$p_{\pi,k}(\ell, m | i, j) = \text{Prob}(E(\mathbb{F}_p)[\pi^{k+1}] \simeq \mathbb{Z}/\pi^\ell\mathbb{Z} \times \mathbb{Z}/\pi^m\mathbb{Z} \mid E_p[\pi^k] \simeq \mathbb{Z}/\pi^i\mathbb{Z} \times \mathbb{Z}/\pi^j\mathbb{Z}).$$

When it is clear from context, π is omitted.

Remark 3.1.16. Since for every natural number m and every prime p coprime to m, $E(\mathbb{F}_p)[m] \subset \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}$, we have $p_{\pi,k}(i,j) = 0$ for $j > k$. In the case $j < k$, if $p_{\pi,k}(\ell, m \mid i, j)$ is defined, it equals 1 if $(\ell, m) = (i, j)$ and equals 0 if $(\ell, m) \neq (i, j)$. Finally, for $j = k$, there are only three conditional probabilities which can be non-zero: $p_{\pi,k}(i, k \mid i, k)$, $p_{\pi,k}(i, k+1 \mid i, k)$, and $p_{\pi,k}(k+1, k+1 \mid k, k)$.

Theorem 3.1.17. *Let π be a prime and E an elliptic curve over $\mathbb{Q}$. If k is an integer such that $I(E, \pi, k+1) = I(E, \pi, k)$, in particular if E has no complex multiplication and $k \geq n(E, \pi)$, then for all $0 \leq i < k$ we have:*

1. $p_{\pi,k}(k+1, k+1 \mid k, k) = \dfrac{1}{\pi^4}$;
2. $p_{\pi,k}(k, k+1 \mid k, k) = \dfrac{(\pi-1)(\pi+1)^2}{\pi^4}$;
3. $p_{\pi,k}(i, k+1 \mid i, k) = \dfrac{1}{\pi}$.

Proof. Let M be the ring $\mathbb{Z}/\pi^{k+1}\mathbb{Z}$. For all $g \in \mathrm{GL}_2(\pi M)$, we consider the set $\text{Lift}(g) = \{h \in \mathrm{GL}_2(M) \mid h_{|\pi M} = g\} = \{g + \pi^k \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \mid a, b, c, d \in \mathbb{Z}/\pi\mathbb{Z}\}$, whose cardinality is π^4. Since $I(E, \pi, k+1) = I(E, \pi, k)$, we have $\dfrac{\#\,\mathrm{Gal}(\mathbb{Q}(E[\pi^k])/\mathbb{Q})}{\#\,\mathrm{Gal}(\mathbb{Q}(E[\pi^{k+1}])/\mathbb{Q})} = \dfrac{\#\,\mathrm{GL}_2(\mathbb{Z}/\pi^k\mathbb{Z})}{\#\,\mathrm{GL}_2(\mathbb{Z}/\pi^{k+1}\mathbb{Z})}$, which equals $\frac{1}{\pi^4}$ by Remark 3.1.6. So for all $g \in \rho_{\pi^k}\left(\mathrm{Gal}(\mathbb{Q}(E[\pi^k])/\mathbb{Q})\right)$, $\text{Lift}(g) \subset \rho_{\pi^{k+1}}(\mathrm{Gal}(\mathbb{Q}(E[\pi^{k+1}])/\mathbb{Q}))$. Thanks to Theorem 3.1.8, the proof will follow if we count for each g the number of lifts with a given fixed group.

1. For $g = \mathrm{Id} \in \rho_{\pi^k}(\mathrm{Gal}(\mathbb{Q}(E[\pi^k])/\mathbb{Q})$, there is only one element of $\text{Lift}(g)$ fixing $(\mathbb{Z}/\pi^{k+1}\mathbb{Z})^2$, so $p_{\pi,k}(k+1, k+1 \mid k, k) = \frac{1}{\pi^4}$.

2. The element $g = \mathrm{Id} \in \rho_{\pi^k}(\mathrm{Gal}(\mathbb{Q}(E[\pi^k])/\mathbb{Q})$, can be lifted in exactly $\pi^4 - 1 - \#\,\mathrm{GL}_2(\mathbb{Z}/\pi\mathbb{Z})$ ways to elements in $\mathrm{GL}_2(\mathbb{Z}/\pi^{k+1}\mathbb{Z})$ which fix the π^k-torsion, a point of order π^{k+1}, but not all the π^{k+1}-torsion.
Therefore $p_{\pi,k}(k, k+1 \mid k, k) = \frac{(\pi-1)(\pi+1)^2}{\pi^4}$.

3. Every element of $\mathrm{GL}_2(\mathbb{Z}/\pi^k\mathbb{Z})$ which fixes a line, but is not the identity, can be lifted in exactly π^3 ways to an element of $\mathrm{GL}_2(\mathbb{Z}/\pi^{k+1}\mathbb{Z})$ which fixes a line of $(\mathbb{Z}/\pi^{k+1}\mathbb{Z})^2$. So $p_{\pi,k}(i, k+1 \mid i, k) = \frac{\pi^3}{\pi^4} = \frac{1}{\pi}$.

□

The theorem below uses the information on $\mathrm{Gal}(\mathbb{Q}(E[\pi^{n(E,\pi)}])/\mathbb{Q})$ for a given prime π in order to compute the probabilities of divisibility by any power of π. We will also compute the average valuation of $\#E(\mathbb{F}_p)$ at π. Let us denote and define it by

$$\overline{v_\pi} = \sum_{k\geq 1} k \,\mathrm{Prob}(v_\pi(\#E(\mathbb{F}_p)) = k),$$

where v_π denotes the valuation at π. One can remark that the sum may have an infinite number of terms, so we must be careful when working with the symbols Prob which denote natural densities.

Notation 3.1.18. Let π be a prime and $\gamma_n(h) = \pi^n \sum_{\ell=0}^{h} \pi^\ell p_n(\ell, n)$. We also define

$$\delta(k) = \begin{cases} p_{i+1}(i+1, i+1) & \text{if } k = 2i+1 \\ 0 & \text{otherwise} \end{cases} \quad \text{and } S_k(h) = \pi^k \left(\sum_{\ell=h}^{\lfloor \frac{k}{2} \rfloor} p_{k-\ell}(\ell, k-\ell) + \delta(k) \right).$$

Theorem 3.1.19. *Let π be a prime, E an elliptic curve over $\mathbb{Q}$, and n be a positive integer such that $\forall k \geq n$, $I(E,\pi,k) = I(E,\pi,n)$ (e.g., a curve without complex multiplication and $n \geq n(E,\pi)$). Then, for any $k \geq 1$,*

$$\mathrm{Prob}(\pi^k \mid \#E(\mathbb{F}_p)) = \begin{cases} \dfrac{S_k(0)}{\pi^k} & \text{if } 1 \leq k \leq n, \\ \dfrac{1}{\pi^k}(\gamma_n(k-n-1) + S_k(k-n)) & \text{if } n < k \leq 2n, \\ \dfrac{1}{\pi^k}(\gamma_n(n) + p_n(n,n)\pi^{2n-1} - \dfrac{\pi^{4n-1} p_n(n,n)}{\pi^k}) & \text{if } k > 2n. \end{cases}$$

Then, $\overline{v_\pi}$ is finite and

$$\overline{v_\pi} = 2\sum_{\ell=1}^{n-1} p_\ell(\ell,\ell) + \frac{\pi}{\pi-1}\sum_{\ell=0}^{n-1} p_n(\ell,n) + \sum_{\ell=0}^{n-2}\sum_{i=\ell+1}^{n-1} p_i(\ell,i) + \frac{\pi(2\pi+1)}{(\pi-1)(\pi+1)} p_n(n,n).$$

Proof. Let k be a positive integer. Using Figure 3.1, one checks that

$$\mathrm{Prob}(\pi^k \mid \#E(\mathbb{F}_p)) = \sum_{\ell=0}^{\lfloor \frac{k}{2} \rfloor} p_{k-\ell}(\ell, k-\ell) + \delta(k). \tag{3.1}$$

Let $c_1 = \frac{1}{\pi^4}$, $c_2 = \frac{(\pi-1)(\pi+1)^2}{\pi^4}$, and $c_3 = \frac{1}{\pi}$. With these notations, the situation can be illustrated by Figure 3.1. For $j > n$ and $\ell < n$, the probability $p_j(\ell, j)$ is the product of the conditional probabilities of the unique path from (ℓ, j) to (ℓ, n) in the graph of Figure 3.1 times the probability $p_n(\ell, n)$. For $j > n$ and $\ell \geq n$, the probability $p_j(\ell, j)$ is the product of the conditional probabilities of the unique path from (ℓ, j) to (n, n) in the graph of Figure 3.1 times the probability $p_n(n, n)$.

There are three cases that are to be treated separately: $1 \leq k \leq n$, $n < k \leq 2n$ and $k > 2n$. For $1 \leq k \leq n$, the result follows from Equation (3.1). Let us explain

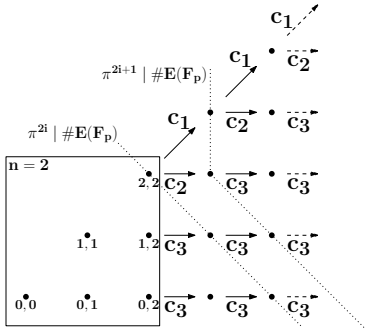


Figure 3.1: Each node of coordinates (i,j) represents the event $\big(E(\mathbb{F}_p)[\pi^j] \simeq \mathbb{Z}/\pi^i\mathbb{Z} \times \mathbb{Z}/\pi^j\mathbb{Z}\big)$. The arrows represent the conditional probabilities of Theorem 3.1.17.

the case for $k > 2n$, with $k = 2i$:

$$\begin{aligned}\mathrm{Prob}(\pi^{2i} \mid \#E(\mathbb{F}_p)) &= \sum_{\ell=0}^{i} p_{2i-\ell}(\ell, 2i-\ell) + \delta(2i) = \sum_{\ell=0}^{i} p_{2i-\ell}(\ell, 2i-\ell) \\ &= \sum_{\ell=0}^{n-1} p_{2i-\ell}(\ell, 2i-\ell) + \sum_{\ell=n}^{i-1} p_{2i-\ell}(\ell, 2i-\ell) + p_i(i,i) \\ &= \sum_{\ell=0}^{n-1} c_3^{2i-l-n} p_n(\ell, n) + \sum_{\ell=n}^{i-1} c_3^{2i-2l-1} c_2 c_1^{l-n} p_n(n,n) + c_1^{i-n} p_n(n,n).\end{aligned}$$

After computations, one obtains the desired formula. The cases $k > 2n$ odd, and $n < k \leq 2n$ are treated similarly.

In order to prove the statements regarding $\overline{v_\pi}$, let us remark that $\mathrm{Prob}(\#E(\mathbb{F}_p) \equiv 0 \mod \pi^k)$ is $O(\frac{1}{\pi^k})$. Indeed for k large enough ($k > 2n$), the proven formula for $\mathrm{Prob}(\#E(\mathbb{F}_p) \equiv 0 \mod \pi^k)$ is bounded by $\frac{C_n}{\pi^k}$, where C_n is a constant depending only on n. This remark proves that $\sum_{k\geq 1} \mathrm{Prob}(\#E(\mathbb{F}_p) \equiv 0 \mod \pi^k)$ is finite. Then, as

$$\mathrm{Prob}(v_\pi(\#E(\mathbb{F}_p)) = k) = \mathrm{Prob}(\#E(\mathbb{F}_p) \equiv 0 \mod \pi^k) - \mathrm{Prob}(\#E(\mathbb{F}_p) \equiv 0 \mod \pi^{k+1})$$

and, by the remark above,

$$\lim_{K\to\infty} K\,\mathrm{Prob}(\#E(\mathbb{F}_p) \equiv 0 \mod \pi^{K+1}) = 0,$$

one shows that

$$\sum_{k\geq 1} k\,\mathrm{Prob}(v_\pi(\#E(\mathbb{F}_p)) = k) = \sum_{k\geq 1} \mathrm{Prob}(\#E(\mathbb{F}_p) \equiv 0 \mod \pi^k).$$

So $\overline{v_\pi}$ is finite and can be computed as $\sum_{k\geq 1} \mathrm{Prob}(\#E(\mathbb{F}_p) \equiv 0 \mod \pi^k)$. If one computes this sum, one obtains the desired formula. □

	Average valuation of 2			Average valuation of 3			Average valuation of 5		
	n	Th.	Exp.	n	Th.	Exp.	n	Th.	Exp.
E_1	1	$\frac{14}{9} \approx 1.556$	1.555	1	$\frac{87}{128} \approx 0.680$	0.679	1	$\frac{695}{2304} \approx 0.302$	0.301
E_3	3	$\frac{895}{576} \approx 1.554$	1.554	1	$\frac{39}{32} \approx 1.219$	1.218	1	$\frac{155}{192} \approx 0.807$	0.807

Table 3.2: Experimental values (Exp.) are obtained with all primes below 2^{25}. Theoretical values (Th.) come from Theorem 3.1.19.

Example 3.1.20. Let us compare the theoretical and experimental average valuation of $\pi = 2$, $\pi = 3$ and $\pi = 5$ for the curve $E_1 : y^2 = x^3 + 5x + 7$ and $E_3 : y^2 = x^3 - 10875x + 526250$. We exclude E_2 in this example since it has complex multiplication. For E_1, we apply Theorem 3.1.19 with $n = 1$ and compute the necessary probabilities with Corollary 3.1.10 knowing that the Galois groups are isomorphic to $\mathrm{GL}_2(\mathbb{Z}/\pi\mathbb{Z})$. For E_3, we apply Theorem 3.1.19 with $n = 3$ for $\pi = 2$ and $n = 1$ for $\pi = 3$ and $\pi = 5$, and compute the necessary probabilities with Corollary 3.1.10 (when $n = 1$) and Theorem 3.1.8 (when $n = 3$). The results are shown in Table 3.2.

In order to apply Theorem 3.1.19, one has to show that $I(E, \pi, k) = I(E, \pi, n)$ for all $k \geq n$ (or $n \geq n(E, \pi)$ since E_1 and E_3 have no complex multiplication). For E_1, we were able to prove that $n(E, \pi) = 1$ for $\pi = 2$, $\pi = 3$ and $\pi = 5$ by using the remarks at the end of section 3.1.2. For E_3, Andrew Sutherland computed for us the Galois groups up to the 2^5-, 3^3- and 5^2-torsion. It is sufficient to compute the probabilities and have some intuition for the values of n, but we were not able to prove that they are correct. In this case, we have to assume that the values of n for which we were able to compute the Galois group (and so the probabilities) are correct.

3.2 Applications to some families of elliptic curves

As shown in the preceding section, changing the torsion properties is equivalent to modifying the Galois group. One can see the fact of imposing rational torsion points as a way of modifying the Galois group. In this section we change the Galois group either by splitting the division polynomials or by imposing some equations that directly modify the Galois group. With these ideas, we find new infinite ECM-friendly families and we explain the properties of some known curves.

3.2.1 Generic Galois group of a family of curves

In the following, when we talk about the *Galois group of the m-torsion of a family of curves*, we talk about a group isomorphic to the Galois group of the m-torsion for all curves of the family except for a sparse set of curves (which can have a smaller Galois group).

For example, let us consider the Galois group of the 2-torsion for the following family $\{\mathcal{E}_r : y^2 = x^3 + rx^2 + x \mid r \in \mathbb{Q}\backslash\{\pm 2\}\}$. The Galois group of the 2-torsion of the curve $\mathcal{E} : y^2 = x^3 + Ax^2 + x$ over $\mathbb{Q}(A)$ is $\mathbb{Z}/2\mathbb{Z}$. Hence, for most values of

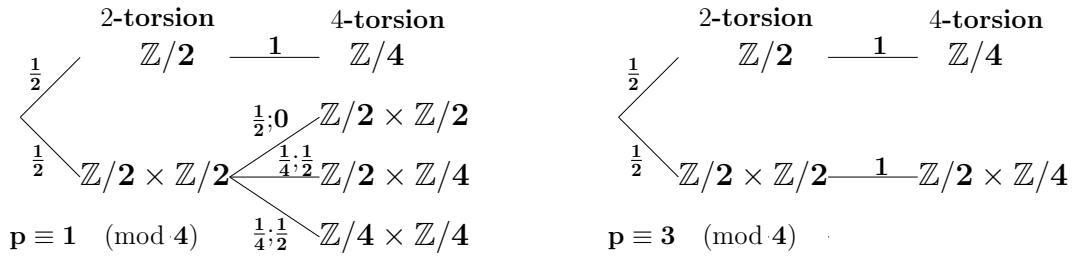


Figure 3.2: Probabilities that $E(\mathbb{F}_p)[m]$ takes each possible value for $m = 2$ and 4. If an edge has a unique probability, the general Montgomery curves and the subfamily satisfying Equation 3.2 are similar. When they have different probabilities we write $p_1; p_2$ where p_1 is the probability in the general case and p_2 in the particular case.

r the Galois group is $\mathbb{Z}/2\mathbb{Z}$ and for a sparse set of values the Galois group is the trivial group. So, we say that the Galois group of the 2-torsion of this family is $\mathbb{Z}/2\mathbb{Z}$.

Let us compute the Galois group for every curve of a family, so that we can guess the Galois group of the family from a finite number of instantiations. In practice, we took a dozen random curves in the family and if all these Galois groups of the m-torsion were the same, we guessed that it was the Galois group of the m-torsion of the family of curves.

Theorem 2.3.18 suggests that, by imposing equations on the parameters a and d of a twisted Edwards curve, we can improve the torsion properties. The case where $\frac{a}{d}$ is a square has been studied in [BBLP13] and [BBL10] for the family of Edwards curves having $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$ (when $a = 1$) respectively $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ (when $a = -1$) rational torsion. Here we focus on two other equations:

$$\exists c \in \mathbb{Q},\ a = -c^2 \qquad (A + 2 = -Bc^2 \text{ for Montgomery curves}), \tag{3.2}$$

$$\exists c \in \mathbb{Q},\ a - d = c^2 \qquad (B = c^2 \text{ for Montgomery curves}). \tag{3.3}$$

The cardinality of the Galois group of the 4-torsion for generic Montgomery curves is 16 and this is reduced to 8 for the family of curves satisfying (3.2). Using Theorem 3.1.8, we can compute the changes of probabilities due to this new Galois group, as illustrated in Figure 3.2. For all curves satisfying (3.2) and all primes $p \equiv 1 \pmod 4$, the probability of having $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ as the 4-torsion group becomes 0 (instead of $\frac{1}{4}$); the probabilities of having $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ and $\mathbb{Z}/4\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ as the 4-torsion group become $\frac{1}{4}$ (instead of $\frac{1}{8}$).

The Galois group of the 8-torsion of the family of curves satisfying (3.3) has cardinality 128 instead of 256 for generic Montgomery curves. Using Theorem 3.1.8, one can see that the probabilities of having an 8-torsion point are improved.

Using Theorem 3.1.19, one can show that both families of curves, the family satisfying (3.2) and the one satisfying (3.3), increase the probability of having the cardinality divisible by 8 from 62.5% to 75% and the average valuation of 2 from $\frac{10}{3}$ to $\frac{11}{3}$.

3.2.2 Better twisted Edwards curves with torsion $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ using division polynomials.

In this section we search for curves such that some of the factors of the division polynomials split and by doing so we try to change the Galois groups. As an example we consider the family of $a = -1$ twisted Edwards curves E_d with $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$-torsion, these curves are exactly the ones with $d = -e^4$ (see [BBL10]). The technique might be used in any context.

Looking for subfamilies.

For a generic d, P_8^{new} splits into three irreducible factors: two of degree 4 and one of degree 16. If one takes $d = -e^4$, the polynomial of degree 16 splits into three factors: two of degree 4, called $P_{8,0}$ and $P_{8,1}$, and one of degree 8, called $P_{8,2}$. By trying to force one of these three polynomials to split, we found four families, as shown in Table 3.3.

$d = -e^4$	"generic" e	$e = g^2$	$e = \frac{2g^2+2g+1}{2g+1}$	$e = \frac{g^2}{2}$	$e = \frac{g-\frac{1}{g}}{2}$
degree of factors of $P_{8,0}$	4	4	4	$2, 2$	$2, 2$
degree of factors of $P_{8,1}$	4	4	4	4	$2, 2$
degree of factors of $P_{8,2}$	8	$4, 4$	$4, 4$	8	8
average valuation of 2	$\frac{14}{3}$	$\frac{29}{6}$	$\frac{29}{6}$	$\frac{29}{6}$	$\frac{16}{3}$
for $p = 3 \bmod 4$	4	4	4	4	5
for $p = 1 \bmod 4$	$\frac{16}{3}$	$\frac{17}{3}$	$\frac{17}{3}$	$\frac{17}{3}$	$\frac{17}{3}$

Table 3.3: Subfamilies of twisted Edwards curves with torsion group isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ and the degrees of the irreducible factors of $P_{8,0}$, $P_{8,1}$ and $P_{8,2}$.

In all these families the generic average valuation of 2 is increased by $\frac{1}{6}$ ($\frac{29}{6}$ instead of $\frac{14}{3}$), except the family $e = \frac{g-\frac{1}{g}}{2}$ for which it is increased by $\frac{2}{3}$, bringing it to the same valuation as for the family of twisted Edwards curves with $a = 1$ and torsion isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$. Note that these four families cover all the curves presented in the first three columns of [BBL10, Table 3.1], except the two curves with $e = \frac{26}{7}$ and $e = \frac{19}{8}$, which have a generic Galois group for the 8-torsion.

The family $e = \frac{g-\frac{1}{g}}{2}$.

In this section, we study in more detail the family $e = \frac{g-\frac{1}{g}}{2}$. Using Theorem 3.1.8 one can prove that the group order modulo all primes is divisible by 16. However, we give an alternative proof which is also of independent interest. We need the following theorem which computes the 8-torsion points that double to the 4-torsion points $(\pm\sqrt[4]{-d^{-1}}, \pm\sqrt[4]{-d^{-1}})$.

Theorem 3.2.1. *Let E_d be a twisted Edwards curve over $\mathbb{Q}$ with $d = -e^4$, $e = \frac{g-\frac{1}{g}}{2}$ and $g \in \mathbb{Q} \setminus \{-1, 0, 1\}$. Let $p > 3$ be a prime of good reduction. If $t \in \{1, -1\}$ such*

that $tg(g-1)(g+1)$ is a quadratic residue modulo p then the points $(x,y) \in E_d(\mathbb{F}_p)$, with $w \in \{1,-1\}$, and

$$x = \pm g^w y, \qquad y = \pm\sqrt{\frac{4tg^{2-w}}{(g-tw)^3(g+tw)}} \tag{3.4}$$

have order eight and double to $(\pm e^{-1}, te^{-1})$.

Proof. Note that all points (x,y) of order eight satisfy $\infty \neq x \neq 0 \neq y \neq \infty$. Following [BBLP13, Theorem 2.10] a point (x,y) doubles to $((2xy : 1+dx^2y^2), (x^2+y^2 : 1-dx^2y^2)) = ((2xy : -x^2+y^2), (x^2+y^2 : 2-(-x^2+y^2)))$. Let $s,t \in \{1,-1\}$ such that (x,y) doubles to (se^{-1}, te^{-1}), hence

$$\frac{2xy}{-x^2+y^2} = \frac{s}{e} \quad \text{and} \quad \frac{x^2+y^2}{2-(-x^2+y^2)} = \frac{t}{e}.$$

From the terms in the first equation we obtain $\left(\frac{x}{y}\right)^2 + \frac{2exs}{y} + e^2 = 1 + e^2$. Write $e = \frac{g-\frac{1}{g}}{2}$ such that $\left(\frac{x}{y} + se\right)^2 = \left(\frac{g+\frac{1}{g}}{2}\right)^2$. Hence $\frac{x}{y} \in \left\{\pm g, \pm\frac{1}{g}\right\}$ depending on the sign s and the sign after taking the square root. This gives $x^2 = G^2y^2$ with $G^2 \in \{g^2, g^{-2}\}$.

From the second equation we obtain $(e-t)x^2 + (e+t)y^2 = 2t$ and substituting x^2 results in $\left((e-t)G^2 + (e+t)\right)y^2 = 2t$. This can be solved for y when $2t\left((e-t)G^2 + (e+t)\right)$ is a quadratic residue modulo p. This is equivalent to checking if any of

$$2t\left((e-1)g^2 + (e+1)\right) = \frac{t(g-1)^3(g+1)}{g}, \tag{3.5}$$

$$2t\left((e-1) + (e+1)g^2\right) = \frac{t(g-1)(g+1)^3}{g} \tag{3.6}$$

is a quadratic residue modulo p. By assumption $tg(g-1)(g+1)$ is a quadratic residue modulo p. Hence, both expression (3.5) and (3.6) are quadratic residues modulo p. Solving for y and keeping track of all the signs results in the formulae in (3.4). □

A direct consequence of this theorem is as follows.

Corollary 3.2.2. *Let $E = E_d$ be a twisted Edwards curve over $\mathbb{Q}$ with $d = -\left(\frac{g-\frac{1}{g}}{2}\right)^4$, $g \in \mathbb{Q} \setminus \{-1,0,1\}$ and $p > 3$ a prime of good reduction. Then $E(\mathbb{Q})$ has torsion group isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ and the group order of $E(\mathbb{F}_p)$ is divisible by 16.*

Proof. We consider two cases.
(1) If $p \equiv 1 \pmod 4$ then -1 is a quadratic residue modulo p. Hence, the 4-torsion points $(\pm i, 0)$ exist (see Figure 2.2) and $16 \mid \#E(\mathbb{F}_p)$.
(2) If $p \equiv 3 \pmod 4$ then -1 is a quadratic non-residue modulo p. Then exactly one of $\{g(g-1)(g+1), -g(g-1)(g+1)\}$ is a quadratic residue modulo p. Using Thm. 3.2.1 it follows that the curve $E(\mathbb{F}_p)$ has eight 8-torsion points and hence $16 \mid \#E(\mathbb{F}_p)$. □

Corollary 3.2.2 explains the good behavior of the curve with $d = -(\frac{77}{36})^4$ and torsion group isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ found in [BBL10]. This parameter can be expressed as $d = -(\frac{77}{36})^4 = -\left(\frac{g-\frac{1}{g}}{2}\right)^4$ for $g = \frac{9}{2}$ and, therefore, the group order is divisible by an additional factor two.

Corollary 3.2.3. *Let $g \in \mathbb{Q} \setminus \{-1, 0, 1\}$, $d = -\left(\frac{g-\frac{1}{g}}{2}\right)^4$ and $p \equiv 1 \pmod 4$ be a prime of good reduction. If $g(g-1)(g+1)$ is a quadratic residue modulo p then the group order of $E_d(\mathbb{F}_p)$ is divisible by 32.*

Proof. All 16 4-torsion points are in $E_d(\mathbb{F}_p)$ (see Figure 2.2). By Thm. 3.2.1 we have at least one 8-torsion point. Hence, $32 \mid \#E_d(\mathbb{F}_p)$. □

We generated different values $g \in \mathbb{Q}$ by setting $g = \frac{i}{j}$ with $1 \leq i < j \leq 200$ such that $\gcd(i, j) = 1$. This resulted in 12 231 possible values for g and Sage [S+11] found 614 non-torsion points. As expected, we observed that they behave similarly as the good curve found in [BBL10].

Parametrization

A family of twisted Edwards curves with $a = -1$, rational torsion group equal to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ and a non-torsion point was presented in [BBL10] using an elliptic parametrization. Using ideas of [BC10], we present here a rational parametrization.

Theorem 3.2.4. *Let $t \in \mathbb{Q} \setminus \{0, \pm 1\}$ and $d = -e^4$, $e = \frac{3(t^2-1)}{8t}$, $x_\infty = (4e^3 + 3e)^{-1}$ and $y_\infty = \frac{9t^4-2t^2+9}{9t^4-9}$. Then the twisted Edwards curve $-x^2 + y^2 = 1 + dx^2y^2$ has torsion group $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ and (x_∞, y_∞) is a non-torsion point.*

Proof. The twisted Edwards curve has torsion group $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ because $d = -e^4$ and e is not equal to 0 and ± 1. The point (x_∞, y_∞) is on the curve and since $x_\infty \notin \{0, \infty, e^{-1}, -e^{-1}\}$ this is a non-torsion point. □

This rational parametrization allowed us to impose additional conditions on the parameter e. For the four families, except $e = g^2$ which is treated below, the parameter e is given by an elliptic curve of rank 0 over $\mathbb{Q}$.

Corollary 3.2.5. *Let $P = (x, y)$ be a non-torsion point on the elliptic curve $y^2 = x^3 - 36x$ having rank 1. Let $t = \frac{x+6}{x-6}$, using notations of Theorem 3.2.4, the curve E_{-e^4} belongs to the family $e = g^2$ and has positive rank over $\mathbb{Q}$.*

3.2.3 Better Suyama curves by a direct change of the Galois group

In this section we will present two families that change the Galois group of the 4- and 8-torsion without modifying the factorization pattern of the 4- and 8-division polynomial.

Suyama-11.

Kruppa observed in [Kru10] that among the Suyama curves, the one corresponding to $\sigma = 11$ finds exceptionally many primes. In 2009 we extended it to an infinite family that we present in detail here.

Experiments show that the $\sigma = 11$ curve differs from other Suyama curves only by its probabilities to have a given 2^k-torsion when reduced modulo primes $p \equiv 1 \pmod 4$. The reason is that the $\sigma = 11$ curve satisfies Equation (3.2). Figure 3.2 illustrates the changes in probabilities of the $\sigma = 11$ curve when compared to curves which do not satisfy Equation (3.2) and shows that Equation (3.2) improves the average valuation of 2 from $\frac{10}{3}$ to $\frac{11}{3}$.

Let us call Suyama-11 the set of Suyama curves which satisfy Equation (3.2). When solving the system formed by Suyama's system plus Equation (3.2), we obtain an elliptic parametrization for σ. Given a point (u, v) on $\mathcal{S}_{11} : v^2 = u^3 - u^2 - 120u + 432$, σ is obtained as $\sigma = \frac{120}{u-24} + 5$. The group $\mathcal{S}_{11}(\mathbb{Q})$ is generated by the points $P_\infty = (-6, 30)$, $P_2 = (-12, 0)$ and $Q_2 = (4, 0)$ of orders ∞, 2 and 2 respectively. We exclude $0, \pm P_\infty$, P_2, Q_2, $P_2 + Q_2$, and $Q_2 \pm P_\infty$, which are the points producing non-valid values of σ. The points $\pm R, Q_2 \pm R$ lead to isomorphic curves. Note that the $\sigma = 11$ curve corresponds to the point $(44, 280) = P_\infty + P_2$.

Edwards $\mathbb{Z}/6\mathbb{Z}$: Suyama-11 in disguise.

In [BBL10, Sec. 5] it is shown that the $a = -1$ twisted Edwards curves with $\mathbb{Z}/6\mathbb{Z}$-torsion over $\mathbb{Q}$ are precisely the curves E_d with $d = \frac{-16u^3(u^2-u+1)}{(u-1)^6(u+1)^2}$ where u is a rational parameter.[2] In particular, according to [BBL10, Sec. 5.3] one can translate any Suyama curve in Edwards language and then impose the condition that $-a$ is a square to obtain curves of the $a = -1$ type. Finally, [BBL10, Sec. 5.5] points out that this family has exceptional torsion properties.

In order to understand the properties of this family, we translate it back to Montgomery language using Theorem 2.3.6. Thus, we are interested in Suyama curves which satisfy equation $A + 2 = -Bc^2$ (the Montgomery equivalent for $-a$ being a square). This is the Suyama-11 family, so its torsion properties were explained in Section 3.2.3. These two families have been discovered independently in our internship report of 2009 and in [BBL10].

Suyama-$\frac{9}{4}$.

In experiments by Zimmermann, new Suyama curves with exceptional torsion properties were discovered, such as $\sigma = \frac{9}{4}$. Further experiments show that their special properties are related to the 2^k-torsion and concern exclusively primes $p \equiv 1 \pmod 4$. Indeed, the $\sigma = \frac{9}{4}$ curve satisfies Equation (3.3) and one can check that its average valuation of 2 is improved from $\frac{10}{3}$ to $\frac{11}{3}$.

Let us call Suyama-$\frac{9}{4}$ the set of Suyama curves which satisfy Equation (3.3). When solving the system formed by Suyama's system plus Equation (3.3), we obtain an elliptic parametrization for σ. Given a point (u, v) on $\mathcal{S}_{9/4} : v^2 = u^3 - 5u$,

[2] There is a typo in the proof of [BBL10, Thm. 5.1]; the $\frac{16u^3(u^2-u+1)}{(u-1)^6(u+1)^2}$ misses a minus sign.

Families	Curves	Average valuation of 2			Average valuation of 3		
		n	Th.	Exp.	n	Th.	Exp.
Suyama	$\sigma = 12$	2	$\frac{10}{3} \approx 3.333$	3.331	1	$\frac{27}{16} \approx 1.688$	1.689
Suyama-11	$\sigma = 11$	2	$\frac{11}{3} \approx 3.667$	3.669	1	$\frac{27}{16} \approx 1.688$	1.687
Suyama-$\frac{9}{4}$	$\sigma = \frac{9}{4}$	3	$\frac{11}{3} \approx 3.667$	3.664	1	$\frac{27}{16} \approx 1.688$	1.687
$\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$	E_{-11^4}	3	$\frac{14}{3} \approx 4.667$	4.666	1*	$\frac{87}{128} \approx 0.680$	0.679
$e = \frac{g-\frac{1}{g}}{2}$	$E_{-\left(\frac{77}{36}\right)^4}$	3	$\frac{16}{3} \approx 5.333$	5.332	1*	$\frac{87}{128} \approx 0.680$	0.679
$e = g^2$	E_{-9^4}	3	$\frac{29}{6} \approx 4.833$	4.833	1*	$\frac{87}{128} \approx 0.680$	0.680
$e = \frac{g^2}{2}$	$E_{-\left(\frac{81}{8}\right)^4}$	3	$\frac{29}{6} \approx 4.833$	4.831	1*	$\frac{87}{128} \approx 0.680$	0.679
$e = \frac{2g^2+2g+1}{2g+1}$	$E_{-\left(\frac{5}{3}\right)^4}$	3	$\frac{29}{6} \approx 4.833$	4.833	1*	$\frac{87}{128} \approx 0.680$	0.679

Table 3.4: Experimental values (Exp.) are obtained with all primes below 2^{25}. The case $n = 1^*$ means that the Galois group is isomorphic to $GL_2(\mathbb{Z}/\pi\mathbb{Z})$.

σ is obtained as $\sigma = u$. The group $\mathcal{S}_{9/4}(\mathbb{Q})$ is generated by the points $P_\infty = (-1, 2)$ and $P_2 = (0, 0)$ of orders ∞ and 2 respectively. We exclude the points $0, \pm P_\infty$, P_2 and $P_2 \pm P_\infty$ which produce non-valid values of σ. If two points in $\mathcal{S}_{9/4}(\mathbb{Q})$ differ by P_2 they correspond to isomorphic curves. We recognize the curve associated to $\sigma = \frac{9}{4}$ when considering the point $(\frac{9}{4}, -\frac{3}{8}) = [2]P_\infty$.

3.2.4 Comparison.

Table 3.4 gives a summary of all the families discussed in this chapter. The theoretical average valuations were computed with Theorem 3.1.19, Theorem 3.1.8 and Corollary 3.1.10 under some assumptions on Serre's exponent (see Example 3.1.20 for more information).

Note that, when we impose torsion points over $\mathbb{Q}$, the average valuation does not simply increase by 1, as can be seen in Table 3.4 for the average valuation of 3.

3.3 Some other applications

Theorem 3.1.8 reduces the search of ECM-friendly elliptic curves to the search of curves with a special Galois group. Hence we replaced the costly experimental evaluation of the average valuation of a curve E at some prime π with the computation of the image of the Frobenius of E at π. Sutherland [Sut12] solves this last problem in a very effective way by using Theorem 3.1.8 in reverse: from the image of the Frobenius modulo small primes one deduces its value.

The first application is to search for elliptic curves with good properties which do not belong to any infinite family of good valuation. For example the CADO-NFS software [BFG+09] uses a small number of elliptic curves, namely 3 in the first version, to test for smoothness. Moreover, one can search for curves with small coefficients because a 1-word times n-word multiplication is faster than a

full n-word times n-word multiplication. Hence, instead of curves of the Suyama family we can use other curves of same properties but which do not verify Suyama's equation.

The second application comes from a conjecture of Montgomery [Mon92]:

Conjecture 3.3.1. *Let E be an elliptic curve with torsion subgroup $\mathbb{Z}/12\mathbb{Z}$ over $\mathbb{Q}$. For any prime p, let $\#E(\mathbb{F}_p)$ denote the order of its reduction modulo p. Then*

a) As p ranges through the primes congruent to 5 modulo 6, the largest power of 3 dividing $\#E(\mathbb{F}_p)$ is 3^α with probability $2 \cdot 3^{-\alpha}$ for each $\alpha \geq 1$.

b) As p ranges through the primes congruent to 3 modulo 4, the largest power of 2 dividing $\#E(\mathbb{F}_p)$ is 2^α with probability $1/4$ if $\alpha = 2$ and probability $3 \cdot 2^{-\alpha}$ if $\alpha \geq 3$.

c) As p ranges through the primes congruent to 5 modulo 8, the largest power of 2 dividing $\#E(\mathbb{F}_p)$ is 2^α with probability $1/4$ if $\alpha = 2$ or $\alpha = 3$, probability $3/16$ if $\alpha = 4$, and probability $5 \cdot 2^{-\alpha}$ if $\alpha \geq 5$.

For every curve that we tested we computed the image of the Frobenius and we obtained the results predicted by Montgomery. The conjecture then states that all the curves with torsion $\mathbb{Z}/12\mathbb{Z}$ over $\mathbb{Q}$ have the same 2^α Galois group for every α. In search of a counter example, Sutherland was able to compute the Frobenius image for $500,000$ elliptic curves. The negative result strengthens the conjecture.

Chapter 4

Improvements to the smoothing problem

In the previous chapter we studied how to improve ECM without playing with the smoothness bounds. In this chapter we put ECM in action to solve a slightly more complex task, called the smoothing problem. This algorithmic task is the bottleneck of the so-called individual logarithm stage of the NFS algorithm, state-of-art in computing discrete logarithms in prime fields (see Chapter 6). In the literature, this problem is usually solved with a naive call to the ECM primitive. We demonstrate that we can do better under the assumption that an ECM-like algorithm is proven. This also gives hints for a practical implementation.
The chapter is organized as follows. After giving a precise description of the problem, we give the direct approach and its performances. We then recall a smoothness result of Pomerance which relies on the theorems in Chapter 1. This gives us a strategy in two steps, that is an improvement to the algorithm. In the last section, we extend this to an arbitrary number of steps to get the best complexity results.

4.1 Exposition of the problem

Most of the discrete logarithm algorithms use an *individual logarithm* stage which computes the desired logarithm using the logarithms computed in previous stages. As before, for a given bound $\mathcal{B}$, we say that an integer is smooth if its prime factors are below $\mathcal{B}$. When computing the discrete logarithm $\log_t s$ in a prime field $\mathbb{F}_p$, one starts the individual logarithm stage by finding an integer h such that

$$(t^h s \bmod p) \text{ is } \mathcal{B}\text{-smooth}$$

for a bound $\mathcal{B}$. Since t is required to be a generator of $\mathbb{F}_p^*$, picking random integers h is equivalent to picking random elements in $[0, p-1]$ and testing their smoothness. This is the case for the Index Calculus and the Number Field Sieve algorithms that we present in Chapter 6. A similar case is that of fields $\mathbb{F}_{p^n}$ with p a prime relatively large when compared to p^n, solved by the Number Field Sieve in High Degree [JLSV06].

The state-of-art algorithm for smoothness testing is ECM, which we presented as a Monte Carlo algorithm. Recall the heuristic analysis of complexity summarized in Fact 2.2.3: for a given bound $\mathcal{B}$ and an integer n, ECM finds the $\mathcal{B}$-smooth part of n with probability $1/2$ in time:

$$L_{\mathcal{B}}(1/2, \sqrt{2})^{1+o(1)}(\log n)^{O(1)}.$$

A similar smoothness test makes use of hyper-elliptic curves [LPP93]. For simplicity we assume first that we have a deterministic algorithm with the same complexity as ECM, but we will later show that a probabilistic algorithm is enough. We can now state this first step of the individual logarithm stage, that we call the smoothing problem, as follows.

Problem 4.1.1 (Smoothing problem). *Assume that one has a black box which, given an integer x less that a parameter n and any bound $\mathcal{B}$, decides if x is $\mathcal{B}$-smooth in time $L_{\mathcal{B}}(1/2, \sqrt{2})^{1+o(1)}(\log n)^{O(1)}$. One is requested a pair of constants $(\alpha, c) \in (0,1) \times (0, \infty)$ and an algorithm with the following characteristics:*

- *the algorithm has as input an integer n and as many terms as needed of a randomly generated sequence of elements in $[0, n]$;*
- *the algorithm outputs a term of the sequence which is $L_n(\alpha, c)$-smooth.*

The problem was already discussed in [Ber04] but with an imposed value of α and c.

We present first the naive algorithm for comparison. We then adapt a commonly used idea when doing selections. One proceeds in two steps: first one reduces the number of candidates by a quick test, then one does a thorough analysis of the remaining candidates, called admissibles, until a good candidate is admitted. For example, some factorization algorithms, see [Pom82], used this idea under the name of early-abort method.

4.1.1 The direct approach

The naive algorithm was first analyzed by Commeine and Semaev [CS06]. Let us call starting smoothness bound a parameter $C = L_n(\alpha, c)$ for positive constants α and c. One considers random integers less than n and tests their C-smoothness using ECM, which we consider deterministic for simplicity of exposition. The number of candidates to be tested is $P_{\text{smooth}}(n, C)^{-1} = L_n\left(1-\alpha, (1-\alpha)/c\right)^{1+o(1)}$. Each test does $L_C(1/2, \sqrt{2})^{1+o(1)}$ operations, since the polynomial factor $(\log n)^{O(1)}$ is hidden by the $o(1)$. By Proposition 1.2.2 this equals $L_n(\alpha/2, \sqrt{2c\alpha})$. Hence we obtain

$$\text{time} = \left(L_n\left(1-\alpha, (1-\alpha)/c\right) L_n\left(\alpha/2, \sqrt{2c\alpha}\right)\right)^{1+o(1)}. \tag{4.1}$$

The optimal values of the parameters are as in the following result.

Theorem 4.1.2 (Commeine & Semaev). *The best value of parameter α in the smoothing problem is $\alpha = 2/3$. When setting $c = 1/\sqrt[3]{3}$, the problem can be solved in time $L_n\left(1/3, \sqrt[3]{3}\right)$.*

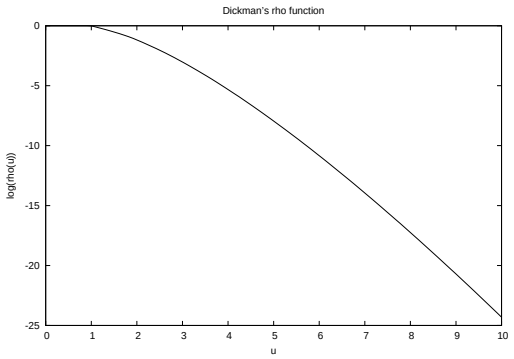


Figure 4.1: Plotting $\log(\rho(u))$ for $u \in [1, 10]$.

4.1.2 Practical improvement

In discrete logarithm algorithms, an improvement [BFHMV84] consists in splitting each candidate into two using a rational reconstruction [vzGG03]. This technique writes any residue z modulo n, coprime to n, as

$$z \equiv z_1 z_2^{-1} \mod n, \tag{4.2}$$

with z_1 and z_2 of order $\sqrt{n}$. Note that this does not fit in the framework of Problem 4.1.1.

The reason for the acceleration might be unclear because we replace $P_{\text{smooth}}(n, C)$ by $P_{\text{smooth}}\left(\sqrt{n}, C\right)^2$, which is roughly equal according to Corollary 1.1.2. Nevertheless, as the function $\log(\rho)$ is concave, where ρ is Dickman's function, the latter probability is greater than the former one. Figure 4.1 shows the graph of $\log(\rho)$ for the values of practical relevance.

If the smoothness probability formula was u^{-u} one would replace u^{-u} by $\left((u/2)^{-u/2}\right)^2$ which is $\exp(u \log 2)$ larger. But, because of the $o(1)$ error in the exponent of the smoothness probability, we need to use the exact statement of Theorem 1.1.1:

$$P_{\text{smooth}}\left(x, x^{\frac{1}{u}}\right) = \exp\left\{-u(\log u + \log\log u - 1 + o(1))\right\}.$$

When $C(n) = L_n(2/3, c)$ a simple computation gives $u = \frac{1}{c}(\log n)^{1/3}(\log\log n)^{-1/3}$ so we have

$$\frac{P_{\text{smooth}}(\sqrt{n}, C)^2}{P_{\text{smooth}}(n, C)} = \exp\left(\frac{\log 2 + o(1)}{c}(\log n)^{1/3}(\log\log n)^{-1/3}\right). \tag{4.3}$$

This is hidden in $L_n(1/3, \cdot)^{o(1)}$, but for the sizes of practical relevance it is an important improvement.

Sieving in the smoothing stage Joux and Lercier [JL03] proposed an improvement to the rational reconstruction idea above. Indeed, the integers z_1 and z_2 in

Equation (4.2) can be computed by a lattice basis reduction as in Algorithm 8.2. Joux and Lercier propose to use the shortest two vectors (z_1, z_2) and (z_1', z_2'), so that we obtain two different fractions congruent to z:

$$z \equiv \frac{z_1}{z_2} \equiv \frac{z_1'}{z_2'} \bmod n. \tag{4.4}$$

Note that $z \equiv (z_1 i + z_1' j)(i z_2 + j z_2')^{-1} \bmod n$ for any integers i and j. Then we can sieve for the pairs (i, j) such that $iz_1 + jz_1'$ and $iz_2 + jz_2'$ are smooth. Unfortunately, the smoothness bound of type $L(2/3)$, makes it impossible to use exclusively the sieve technique. One can use sieving as a first step which selects pairs (i, j) with a large B'-smooth part, where B' is a parameter of type $L(1/3)$. In the light of the admissibility strategy, we use the sieve as an admissibility test. The main test is a smoothness test with a parameter of size $L(2/3)$.

4.2 Stronger smoothness results

Let us present some preparatory results for the new strategy. First, it is intuitive that larger numbers have a smaller smoothness probability. The technicality is to say that the $o(1)$ error is the same for two values of x which are sufficiently large.

Lemma 4.2.1. *Let α be a positive constant. For all $\delta > 0$, there exists $x_\delta > 0$ such that, for all reals x, x' and y such that $x \geq x_\delta$ and $y^{1+\alpha} \leq x, x' \leq \exp(y^{1-\alpha})$ we have*

$$x \leq x' \Rightarrow P_{\text{smooth}}(x, y) \leq P_{\text{smooth}}(x', y)^{1-\delta}.$$

Proof. Take $\epsilon > 0$ sufficiently small such that $2\epsilon/(1-\epsilon) < \delta$. Since we supposed $x \leq x'$, if we put $u' = \log x'/\log y$ and $u = \log x/\log y$, then $u' \geq u$. Using Theorem 1.1.1, for sufficiently large x and x', we obtain

$$P_{\text{smooth}}(x', y) \leq (u')^{-u'(1+\epsilon)} \leq u^{-u(1+\epsilon)} \leq P_{\text{smooth}}(x, y)^{1-\bar{\epsilon}},$$

where $\bar{\epsilon} = 2\epsilon/(1-\epsilon)$. Since $\bar{\epsilon} < \delta$ we get the result. □

Pomerance [Pom82] strengthened Theorem 1.1.1 when studying the early abort method in some factorization algorithms. Let us first define $\Psi(x, y, z)$ as the set of positive integers less than x for which all the prime factors are in the interval $[z, y]$, in particular $\Psi(x, y, 1) = \Psi(x, y)$. We denote the cardinality of $\Psi(x, y, z)$ by $\psi(x, y, z)$ and we call smoothness probability the quantity $P_{\text{smooth}}(x, y, z) = \psi(x, y, z)/x$.

Theorem 4.2.2 ([Pom82], Theorem 2.2)**.** *Let $\alpha > 0$. For all $\epsilon > 0$ there exists $x_\epsilon > 0$ such that for all triples x, y, z such that*

$$x \geq x_\epsilon, \quad (\log x)^{1+\alpha} \leq y \leq \exp\left((\log x)^{1-\alpha}\right), \quad z \leq y^{1-1/\log u},$$

where $u = \log x/\log y$, one has

$$u^{-u(1-\epsilon)} \leq P_{\text{smooth}}(x, y, z) \leq u^{-u(1+\epsilon)}.$$

As the theorem has a complex statement, it is preferable to restate two of its cases of application as a corollary. Next, we could obtain a version of Theorem 4.2.2 in L notation as we did with Theorem 1.1.1, but we restrict ourselves to two very specific situations.

Corollary 4.2.3. *Let $a, c, \theta, \theta' > 0$ be four real numbers with $\theta' < \theta$. For all positive integer n we set $L(n) = L_n(2/3, a)$, which for ease of notations is also written L. Then we have*

$$\begin{aligned} P_{\text{smooth}}\left(n^c, L^\theta, L^{\theta'}\right) &= L_n\left(1/3, \frac{c}{3\theta a}\right)^{-1+o(1)} \\ P_{\text{smooth}}\left(n^c L_n(1/2,1), L^\theta, L^{\theta'}\right) &= L_n\left(1/3, \frac{c}{3\theta a}\right)^{-1+o(1)} \end{aligned}$$

Proof. We compute $u(n) = \log(n^c)/\log(L(n)^\theta)$ or $u(n) = \log\left(n^c L_n(1/2,1)\right)/\log(L(n)^\theta)$. In both cases we obtain

$$u = \frac{c \log n}{a\theta \log n^{2/3} (\log\log n)^{1/3}} = \frac{c}{\theta a}\left(\frac{\log n}{\log\log n}\right)^{1/3}. \tag{4.5}$$

Hence $u^u = L_n\left(1/3, \frac{c}{3a\theta}\right)$, which gives the result. □

4.3 Selection with one admissibility test

We saw that, when solving Problem 4.1.1, the optimal value of α is $2/3$ (Theorem 4.1.2). Hence, we search for elements which are $L(n)$-smooth, where $L(n) := L_n(2/3, a)$ for a parameter a to be chosen. Let us consider as admissibility test the utilization of the black box on the input candidates with a smoothness bound $L(n)^\theta$ for a parameter $\theta < 1$ to be chosen. A candidate is admissible if the admissibility test finds a factor of size at least n^c for a parameter c to be chosen. The set of admissible candidates will be denoted by M_1, while the set of admissible candidates which succeed the main test is denoted M_2. To show that this is a good admissibility test we show that it drastically reduces the number of candidates, but that, among the admissible candidates, there is at least one which will pass the main test.

Theorem 4.3.1. *Let a be a real. For any integer n put $L(n) = L_n(2/3, a)$, also denoted L. Let c and θ be two reals in the interval $(0,1)$. Call M_1 the set of positive integers less than n whose L^θ-smooth part is larger than n^c. Call M_2 the subset of M_1 formed of these elements which are L-smooth. Then we have:*

(i) $\#M_1 \leq nL_n\left(1/3, \frac{c}{3a\theta}\right)^{-1+o(1)}$;

(ii) $\#M_2 \geq nL_n\left(1/3, \frac{c}{3\theta a} + \frac{1-c}{3a}\right)^{-1+o(1)}$.

Proof. :
(i) Any element of M_1 can be uniquely written as the product of its L^θ-smooth part m and its other divisor t. Put $T = \Psi(n^{1-c}, n^{1-c}, L^\theta) \bigcup \{1\}$. We split M_1 in disjoint subsets having the same divisor t in T as follows

$$M_1 = \bigcup_{t \in T} \left\{ tm \mid m \in \Psi(n/t, L^\theta), m \geq n^c \right\}. \tag{4.6}$$

Hence $\#M_1 \leq \sum_{t \in T} \psi(n/t, L^\theta)$.

Let $\epsilon > 0$ and x_ϵ as in Corollary 4.2.1. Then, for n sufficiently large so that $n^c \geq x_\epsilon$, we have

$$\forall t \in T,\ P_{\text{smooth}}\left(n/t, L^\theta\right) \leq P_{\text{smooth}}\left(n^c, L^\theta\right)^{1-\epsilon}. \tag{4.7}$$

Hence, for all t in T, we have

$$\psi\left(n/t, L^\theta\right) \leq (n/t) P_{\text{smooth}}\left(n^c, L^\theta\right)^{1-\epsilon}. \tag{4.8}$$

When injecting in Equation 4.6 we obtain

$$\#M_1 \leq n \left(\sum_{t \in T} 1/t \right) P_{\text{smooth}}\left(n^c, L^\theta\right)^{1-\epsilon}. \tag{4.9}$$

Using Corollary 4.2.3, for large enough n we have

$$P_{\text{smooth}}\left(n^c, L^\theta\right) \leq L_n\left(1/3, \frac{c}{3\theta a}\right)^{1-\epsilon}. \tag{4.10}$$

Also, the sum of terms $1/t$ in T is dominated by the sum $1/t$ when t goes from 1 to n, which is polynomial in $\log n$, so it is dominated by $L_n(1/3, \cdot)^\epsilon)$ for any constant at the place of $\cdot$. We obtain the announced result

$$\#M_1 \leq n L_n\left(1/3, \frac{c}{3\theta a}\right)^{-(1-2\epsilon)}. \tag{4.11}$$

(ii) The basic idea is that any element in M_2 can be written as $t \times m$ with t an element of $\Psi(n^{1-c}, L, L^\theta)$ and m roughly in $\Psi(n^c, L^\theta)$. The difficulty we face is that m is supposed larger than n^c, so we have to extend slightly the set of elements m.

Put $\ell(n) = L_n(1/2, 1)$. Clearly, any product $t \times m$ with m in $\Psi(n^c \ell(n), L^\theta) \backslash \Psi(n^c, L^\theta)$ and $t \in \Psi(n^{1-c}/\ell(n), L, L^\theta)$ belongs to M_2. Moreover, two different pairs (t, m) cannot have the same product. Hence we have

$$\#M_2 \geq \left[\psi\left(n^c \ell(n), L, L^\theta\right) - \psi\left(n^c, L, L^\theta\right) \right] \psi\left(n^{1-c}/\ell(n), L, L^\theta\right). \tag{4.12}$$

By Corollary 4.2.3 we have $P_{\text{smooth}}(n^c, L^\theta) = P_{\text{smooth}}(n^c\ell(n), L^\theta)^{1+o(1)}$. Hence the value in the bracket, call it B, is

$$\begin{aligned} B &= n^c\ell(n)L_n\left(1/3, -\frac{c}{3\theta a}\right)\left[1 - L(1/3,1)^{o(1)}/\ell(n)\right] \\ &= n^c\ell(n)L_n\left(1/3, -\frac{c}{3\theta a}\right)^{1+o(1)}. \end{aligned}$$

Again Corollary 4.2.3 gives $\psi(n^{1-c}/\ell(n), L, L^\theta) = n^{1-c}/\ell(n)L_n\left(1/3, -\frac{1-c}{3a}\right)^{1+o(1)}$. This completes the proof. □

Optimal parameters

Let us optimize the parameters for finding one admitted number, i.e., finding an element of M_2. Note first that each admissibility test costs $t_1 := L_n\left(1/3, \sqrt{4a\theta/3}\right)$ and each main test costs $t_{\text{main}} := L_n\left(1/3, \sqrt{4a/3}\right)$. The total number of candidates to be tested before finding one in M_2 is

$$\text{number of candidates} = \frac{\#M_2}{n}. \tag{4.13}$$

The proportion of candidates which succeed the admissibility test is $n/\#M_1$, so

$$\text{number of admissibles} = \frac{\#M_2}{\#M_1}. \tag{4.14}$$

Thus, the average time for the selection of one element in M_2 is:

$$\text{time (selection)} = \frac{n}{\#M_2}t_1 + \frac{\#M_1}{\#M_2}t_{\text{main}}, \tag{4.15}$$

which, according to Theorem 4.3.1, is dominated by:

$$L_n\left(\frac{1}{3}, \frac{1-c}{3a} + \frac{c}{3\theta a} + \sqrt{\frac{4}{3}\theta a}\right)^{1+o(1)} + L_n\left(\frac{1}{3}, \frac{1-c}{3a} + \sqrt{\frac{4}{3}a}\right)^{1+o(1)}. \tag{4.16}$$

For fixed values of a and c, we minimize the time by setting $\theta = \left(\frac{c^2}{3a^3}\right)^{\frac{1}{3}}$. This leads to a complexity as follows:

$$L_n\left(\frac{1}{3}, \frac{1-c}{3a} + 3\left(\frac{c}{9}\right)^{\frac{1}{3}}\right)^{1+o(1)} + L_n\left(\frac{1}{3}, \frac{1-c}{3a} + \sqrt{\frac{4}{3}a}\right)^{1+o(1)}. \tag{4.17}$$

For a fixed value of a, we minimize this expression by setting $c = \frac{(\frac{4}{3}a)^{\frac{3}{2}}}{3}$. We obtain a function in a which reaches its minimum for $a \approx 0.7715$. It gives $c \approx 0.348$ and $\theta \approx 0.445$, which verify the condition $0 < \theta, c < 1$. We insert the numerical values in (4.16) and obtain:

$$\text{time (selection)} = L_n\left(\frac{1}{3}, 1.296\right)^{1+o(1)}. \tag{4.18}$$

This improves on the complexity $L_n(1/3, 1.442)^{1+o(1)}$ of the direct approach.

4.4 The admissibility strategy

Encouraged by our success with one admissibility test, we can imagine a strategy with $k \geq 2$ admissibility tests, taken one after another. Let $0 < \theta_1 < \cdots < \theta_k < 1$ be k parameters. Also let $0 < c_1, \ldots, c_k < 1$ be k parameters such that $c_1 + \cdots + c_k \leq 1$. For convenience we put $\theta_0 = 0$. A natural generalization of the admissibility idea is as follows. For $h \in [1, k]$, we say that a candidate number m succeeds the h^{th} admissibility test if it is $(h-1)^{\text{th}}$ admissible and m has a divisor m_h larger that m^{c_h} all of whose prime factors are in the interval $[L(n)^{\theta_{h-1}}, L(n)^{\theta_h}]$. Each candidate takes the first admissibility test. If it succeeds, it goes on to the next admissibility test, whereas if it fails we discard it and consider the next candidate.

We analyzed this strategy in our Master Thesis in 2011. Here we present a slightly different strategy which allows us to have a simpler presentation, without obtaining a worse complexity. Put $\ell(n) := L_n(1/2, 1)$. For any h in $[1, k]$, we say that a candidate is h^{th} admissible if it has h factors m_1, ..., m_h such that, for $i \in [1, h]$,

$$m_i \text{ belongs to } \mathcal{B}_i := \Psi\left(n^{c_i}\ell(n), L^{\theta_i}, L^{\theta_{i-1}}\right) \setminus \Psi\left(n^{c_i}, L^{\theta_i}, L^{\theta_{i-1}}\right). \tag{4.19}$$

We call M_h the set of h^{th} admissible candidates and $\overline{M}$ the set of elements of M_k which succeed the main test, i.e., which are L-smooth.

The following result generalizes theorem 4.3.1.

Theorem 4.4.1. *Let $a > 0$ be a real. For all n we put $L(n) = L_n(2/3, a)$. Then, for M_1, ..., M_k and $\overline{M}$ defined above, we have:*

(i) $\#\overline{M} \geq n \cdot L_n\left(1/3, \frac{c_1}{3\theta_1 a} + \cdots + \frac{c_k}{3\theta_k a} + \frac{1-(c_1+\cdots+c_k)}{3a}\right)^{-1+o(1)}$;

(ii) $\forall h \in [1, k], \#M_h \leq n \cdot L_n\left(1/3, \frac{c_1}{3\theta_1 a} + \cdots + \frac{c_h}{3\theta_h a}\right)^{-1+o(1)}$.

Proof. (i) Every element m of $\overline{M}$ can be written as

$$m = m_1 \cdots m_k t$$

with $m_i \in \mathcal{B}_i$ and $t \in \Psi(n^{1-c_1-\cdots-c_k}, L, L^{\theta_k})$. We put $B_i = \psi(n^{c_i}\ell(n), L^{\theta_i}, L^{\theta_{i-1}}) - \psi(n^{c_i}, L^{\theta_i}, L^{\theta_{i-1}})$. As a lower bound for the set of values of t which occur in elements m of M_h we use $\overline{B} := \psi(n^{1-c_1-\cdots-c_k}/\ell(n)^k, L, L^{\theta_k})$, so one has $\#\overline{M} \geq (\prod_{i=1}^{k} B_i)\overline{B}$. One also has

$$B_i = n^{c_i}\ell(n) P_{\text{smooth}}(n^{c_i}\ell(n), L^{\theta_i}, L^{\theta_{i-1}}) \left(1 - \ell(n)^{-1} \frac{P_{\text{smooth}}\left(n^{c_i}, L^{\theta_i}, L^{\theta_{i-1}}\right)}{P_{\text{smooth}}\left(n^{c_i}\ell(n), L^{\theta_i}, L^{\theta_{i-1}}\right)}\right). \tag{4.20}$$

By Corollary 4.2.3, each of the smoothness probabilities equal $L_n(1/3, \frac{c_i}{3a\theta_i})^{-1+o(1)}$. In particular, the ratio of the probabilities in the parenthesis equals $L_n(1/3, 1)^{o(1)}$, which is dominated by $\ell(n) = L_n(1/2, 1)$. Hence the parenthesis equals $1 + o(1)$. We obtain

$$B_i = n^{c_i}\ell(n) L_n\left(1/3, \frac{c_i}{3a\theta_i}\right)^{-1+o(1)}. \tag{4.21}$$

We use again Corollary 4.2.3 and obtain $\overline{B} = \frac{n^{1-c_1-\cdots-c_k}}{\ell(n)^k} L_n(1/3, \frac{1-(c_1+\cdots+c_k)}{3a})^{1+o(1)}$. This completes the proof of (i).

(ii) Let $N(n)$ be a function which depends on n, that we specify later. Every element m of M_h can be written as $m = m_1 \cdots m_h t_h$ with m_i in $\mathcal{B}_i$. Consider a tuple of indices $j_1,\ldots,j_h$ with $j_i \in [0, N-1]$ and consider the case when all $m_i \in \mathcal{B}_i \bigcap [n^{c_i}\ell(n)^{j_i/N}, n^{c_i}\ell(n)^{(j_i+1)/N}]$. We have $\mathcal{B}_i \bigcap [n^{c_i}\ell(n)^{j_i/N}, n^{c_i}\ell(n)^{(j_i+1)/N}] \subset \Psi(n^{c_i}\ell(n)^{(j_i+1)/N}, L^{\theta_i})$, so

$$\begin{aligned} \#\mathcal{B}_i \bigcap [n^{c_i}\ell(n)^{j_i/N}, n^{c_i}\ell(n)^{(j_i+1)/N}] &\leq \psi(n^{c_i}\ell(n)^{(j_i+1)/N}, L^{\theta_i}) \\ &= n^{c_i}\ell(n)^{(j_i+1)/N} P_{\text{smooth}}(n^{c_i}\ell(n)^{(j_i+1)/N}, L^{\theta_i}). \end{aligned}$$

Due to Lemma 4.2.1 the smoothness probability above equals $P_{\text{smooth}}(n^{c_i}, L^{\theta_i})^{1+o(1)}$.

Since $t_h = m/\prod_{i=1}^{h} m_i$, we have

$$t_h \in [n^{1-c_1-\cdots c_h}/\ell(n)^{(h+j_1+\cdots+j_h)/N}, n^{1-c_1-\cdots c_h}/\ell(n)^{(j_1+\cdots+j_h)/N}].$$

Hence, for each tuple $(j_1, \ldots, j_h)$ we have

$$\#\{m \in M_h \mid m = m_1 \cdots m_h t_h, m_i \in \mathcal{B}_i \bigcap [n^{c_i}\ell(n)^{j_i/N}, n^{c_i}\ell(n)^{(j_i+1)/N}]\} \leq n\ell(n)^{h/N} \prod_{i=1}^{h} P_{\text{smooth}}(n^{c_i}, L^{\theta_i})^{1+o(1)},$$

where the $o(1)$ can be made independent of the tuple $(j_1, \ldots, j_h)$ and of N.

When we sum the inequalities above for all the N^h tuples we obtain:

$$\#M_h \leq (N^h \ell(n)^{h/N}) n \prod_{i=1}^{h} P_{\text{smooth}}(n^{c_i}, L^{\theta_i})^{1+o(1)}. \tag{4.22}$$

We take $N = \log n$ so that $N^h \ell(n)^{h/N} = L(1/3, 1)^{o(1)}$. By Corollary 4.2.3 we can evaluate the smoothness probabilities in Equation (4.22) and obtain the result. □

Note that the $\ell(n)$ trick allowed us to simplify the proof of the (i) part of the theorem whereas one can use the same proof for a lower bound in part (ii).

4.5 Best parameters in the admissibility strategy

Let us first note that, for each h, the h^{th} admissibility test done with ECM takes time $t_h := L_n\left(1/3, \sqrt{4a\theta_h/3}\right)$, whereas the main test takes time $t_{\text{main}} := L_n\left(1/3, \sqrt{4a/3}\right)$. For convenience we put $M_0 = [1, n]$. The total number of candidates that must be tested before finding one in $\overline{M}$ is as follows

$$\text{number of candidates} = n/\#\overline{M}. \tag{4.23}$$

For any $h \in [1, k]$, the proportion of candidates which succeed the h^{th} admissibility test is $\#M_h/n$. Hence we have

$$\text{number of } h^{\text{th}} \text{ admissible candidates} = \frac{\#M_h}{\#\overline{M}}. \tag{4.24}$$

Therefore, the time to find an element of $\overline{M}$ is

$$\text{time (selection)} = \left(\sum_{h=1}^{k} \frac{\#M_{h-1}}{\#\overline{M}} t_h \right) + \frac{\#M_k}{\#\overline{M}} t_{\text{main}}. \tag{4.25}$$

Hence we have time(selection) $= L_n\left(\frac{1}{3}, \max\{f_1, \ldots, f_{k+1}\}\right)^{1+o(1)}$, where

$$f_i := \left(\frac{c_i}{3\theta_i a} + \cdots + \frac{c_k}{3\theta_k a} \right) + \frac{1 - (c_1 + \cdots + c_k)}{3a} + \sqrt{\frac{4}{3}\theta_i a}, \tag{4.26}$$

for $i \in [1, k]$ and $f_{k+1} = \frac{1-(c_1+\cdots+c_k)}{3a} + \sqrt{\frac{4}{3}a}$. We put $\theta_{k+1} = 1$ since it unifies the definitions for f_i with $i \in [1, k]$ and for f_{k+1}. For fixed values of $\theta_{i+1}, \ldots, \theta_{k+1}, c_1, \ldots, c_k$, we minimize f_i by imposing the following relation between c_i and θ_i:

$$\theta_i = \left(\frac{c_i^2}{3a^3} \right)^{\frac{1}{3}}. \tag{4.27}$$

We also impose $f_i = f_{i+1}$ by setting

$$c_i = 9 \left(\frac{\omega}{3} \right)^3 \text{ with } \omega = \left(\frac{4}{3} a \theta_{i+1} \right)^{\frac{1}{2}}. \tag{4.28}$$

This allows us to compute in order c_k, θ_k, c_{k-1}, $\cdots$ and to eliminate the variables c_i and θ_i from Equation 4.26. Hence, we have to minimize an expression in a alone.

For each $k \in [0, 6]$ we computed the optimal values of a and the associated values of the parameters $(c_1, \theta_1, \ldots, c_k, \theta_k)$. Note that Equation (4.28) imposes for the parameter c_k a value which is independent of k. Similarly Equations (4.28) and (4.27) impose that, for each i, the value of c_i in $(c_1, \theta_1, \cdots, c_k, \theta_k)$ depends on $k - i$ rather than on the pair (k, i).

We summarized in Table 4.1 the best values for the parameters a and θ_i, c_i for $i \in [0, 6]$. Note that the case $k = 0$, i.e., without admissibility test, corresponds to the case studied in Section 4.1.1. Finally, we conclude that a sensible choice is to make an admissibility strategy selection with 6 tests. In this case we obtain

$$\text{time(strategy selection)} = L_n\left(\frac{1}{3}, 1.232\ldots \right)^{1+o(1)}. \tag{4.29}$$

In order to use the answer we gave to Problem 4.1.1 we recall that ECM is a Monte Carlo algorithm which tests smoothness. Since each candidate takes at most 6 tests and in at least $1/2^6$ cases all the answers of ECM are correct, we conclude that ECM plays the role of a black box for our problem. The following result assumes that the Fact 2.2.3 can be proved.

Theorem 4.5.1. *Assume the existence of an algorithm which tests the $\mathcal{B}$-smoothness of any element in $[1, n]$ in time $L_{\mathcal{B}}(1/2, \sqrt{2})^{1+o(1)}(\log n)^{O(1)}$, has a 50% probability of correct answer if the candidate is smooth and always gives the correct answer when the candidate is not smooth. Then there exists an algorithm which, given a randomly generated sequence, computes a $L_n(2/3, 0.811)$-smooth element of the sequence in $L_n(1/3, 1.232)^{1+o(1)}$ operations.*

k	0	1	2	3	4	5	6	7
a	0.693	0.771	0.799	0.808	0.811	0.811	0.811	0.811
time	1.442	1.296	1.251	1.238	1.234	1.233	1.232	1.232

$k-i$	0	1	2	3	4	5	6
c_i	0.367	0.109	0.032	0.010	0.003	0.001	≈ 0.000
θ_i	0.444	0.198	0.088	0.039	0.017	0.008	0.003

Table 4.1: Values of the parameters that occur in the admissibility strategy.

Part II

Discrete logarithms in finite fields

Chapter 5

Basic discrete logarithm algorithms

This first chapter of the second part is a warm-up before the more complicated topic of NFS and FFS that we will see in the next chapters. We concentrate here on algorithms which do not require mathematical background on number and function fields. Some generic algorithms can be used in the particular for finite fields and are used in the nowadays algorithms as prerequisites which drop some of the technicalities. Our wish is to give the general ideas about the main complexities that we encounter: $L(1/2, \cdot)$ for Index Calculus algorithms and $L(1/3, \cdot)$ for modern algorithms, that we illustrate with Joux-Lercier's "two rational side FFS".
We organize the chapter as follows. After presenting the generic algorithms, we introduce Index Calculus, the first sub-exponential algorithm for finite fields. It has a complexity of $L(1/2, \cdot)$ and introduces the general lines of all sieving algorithms. Then we show how, by using the similarity between numbers and polynomials, Index Calculus can be translated to the case of finite fields of small characteristic. We conclude the chapter by presenting an algorithm of complexity $L(1/3, \cdot)$.

5.1 Generic algorithms for discrete logarithm

Given two elements t and s of a cyclic group G such that t is a generator, the Discrete Logarithm Problem (DLP) consists in computing the smallest non-negative integer x such that

$$t^x = s. \tag{5.1}$$

We denote x by $\log_t s$ and note that if t' is a second generator then $\log_t(s) = \log_{t'}(s) \log_t(t') \bmod \#G$. Some authors define the DLP as the resolution of Equation (5.1) for arbitrary values of t and s so that there might or might not exist a solution. In cryptography the cardinality of the discrete logarithm group is assumed known as well as a generator. In this work we assume that we know the cardinality and that we know its factorization, except maybe for some very large factors. For simplicity we assume that we have a generator t of the discrete logarithm group, but this is not necessary for the algorithms.

Pohlig-Hellman An easy but important reduction is that, if one has a black box which solves the DLP in cyclic groups of prime order, then one can solve it in any group by the technique of Pohlig and Hellman [PH78]. Indeed, suppose that the factorization of $\#G$ is known: $\#G = \prod_{i=1}^{k} p_i^{e_i}$ for prime numbers $p_1, \ldots, p_k$ and exponents $e_1, \ldots, e_k$. If one computes, for each i a value x_i such that $\left(t^{\#G/p_i^{e_i}}\right)^{x_i} = s^{\#G/p_i^{e_i}}$, then one obtains x by the Chinese Remainder Theorem. Hence we reduced the problem to the case $\#G = p^e$. Call $a_0, \ldots, a_{e-1}$ the digits of the base-p expansion of $\log_t s$: $\log_t s = a_0 + \cdots + a_{e-1}p^{e-1}$ with $a_0, \ldots, a_{p^{e-1}}$ in $[0, p-1]$. We compute a_0, ..., a_{e-1} by a series of DLP instances in groups of prime order, as follows. One has

$$s^{p^{e-1}} = t^{a_0 p^{e-1}},$$

and therefore $a_0 = \log_{t_0} s_0$ for $t_0 = t^{p^{e-1}}$ and $s_0 = s^{p^{e-1}}$. Since t_0 and s_0 belong to a subgroup of order p, we reduced the computation of a_0 to that of solving DLP in a group of prime order. Recursively, we can compute $a_1 + a_2 p + \cdots + a_{e-1}p^{e-2}$ as the logarithm of $(t^{-a_0}s)^p$ in base t^p.

Baby-step giant-step Shanks proposed a method to solve the DLP in any cyclic group G in $O(\sqrt{\#G})$ operations. Let t be a generator of G and s any element of G; then put $n = \lceil\sqrt{\#G}\rceil$. We make two tables storing (i, t^{in}) for $i \in [0, n]$ and (j, st^{-j}) for $j \in [0, n]$. Note that, if one finds i and j such that $t^{in} = st^{-j}$, then one obtains $\log_t s = in + j$. Hence, all we have to do is to find a match between the two tables, which can be done in $\widetilde{O}(\sqrt{\#G})$ operations. In practice, one can use hash-tables and have a time proportional to $\sqrt{\#G}$.

Pollard's rho Shank's method was later overpassed by Pollard's rho method with the same time complexity but with no memory usage (see [Pol78]). It starts by choosing two maps $f, g : [0, \#G - 1] \to [0; \#G - 1]$ which look random. For example in practice one can use $x \mapsto (x^2 + c \bmod \#G)$ where c is a randomly chosen constant [Tes01]. Then one computes two sequences a_i and b_i given by $a_{i+1} = f(a_i)$ and $b_{i+1} = g(b_i)$. At the same time one computes sequences $a_{2i} = f\left(f(a_{2i-2})\right)$ and $b_{2i} = g\left(g(b_{2i-2})\right)$. Given a_i and b_i we can compute in polynomial time x_i defined as follows

$$x_i = t^{a_i} s^{b_i}. \tag{5.2}$$

The sequence x_0, x_1, ... is the orbit of an element by a random permutation of G. This kind of sequences describe ρ-shaped orbits: after a finite number of steps–the so-called tail–, they loop on a circle as in Figure 5.1. Using similar ideas as the Birthday paradox theorem one can expect that the average period of an orbit of this kind is $O(\sqrt{\#G})$. If we call T this period, when i is a multiple of T larger than the cardinality of the tail we have $x_{2i} = x_i$ or equivalently

$$t^{a_i} s^{b_i} = t^{a_{2i}} s^{b_{2i}}. \tag{5.3}$$

With a probability of $1 - o(1)$ we have $b_i \neq b_{2i}$. Then we directly obtain $\log_t s = (a_{2i} - a_i)(b_i - b_{2i})^{-1} \bmod \#G$.

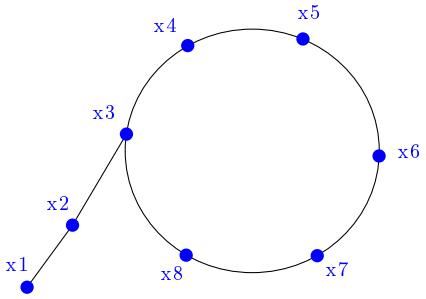


Figure 5.1: Typical orbit of an element by a random permutation.

Consequences in cryptography Due to Pohlig-Hellman's method the groups whose orders are smooth are forbidden for DLP-based crypto-systems. For example if one searches a prime such that the DLP in $\mathbb{F}_p^*$ is secure, one will check that $p-1$ has a large prime factor π. If π is larger than $\sqrt{p}$ we know that π^2 cannot divide $p-1$. In this thesis, all the discrete logarithms are computed modulo such primes π. Also, the condition that t is a generator is replaced by the condition that the order of t is a multiple of π, which happens with probability $1-1/\pi$.

Due to Pollard's rho method, if one desires a security of 2^{128} elementary operations, one must use a group of at least 2^{256} elements. As Pollard's rho is the state-of-art algorithm for generic elliptic curves, it is an active research topic to reduce the constant hidden in the $O()$ [Tes01].[GPR13],[BL12].

5.2 Index Calculus

5.2.1 The algorithm

Many DLP algorithms known today share the general lines of Index Calculus. Given a prime p, Index Calculus starts by computing the discrete logarithm of small primes, seen as element of $\mathbb{F}_p^*$; this part of the algorithm is called *main phase*. In the second part of the algorithm, called *individual logarithm*, one computes any logarithm $\log_t s$ by expressing it with respect to the logarithms computed in the first part.

More precisely, let p be a prime and let us consider a prime divisor π of $p-1$. Let t be an element of order π in $\mathbb{F}_p^*$ and s an other element of $(\mathbb{Z}/p\mathbb{Z})^*$ whose discrete logarithm is sought. We choose B to be the closest integer to $L_p(1/2,\beta)$ for a constant β specified later. We call factor base the set of primes less than B: $\mathcal{F}(B) = \{q \text{ prime} \mid q < B, q \neq t\}$. The *main phase* goes as in Algorithm 5.1.

Note that the smoothness test in line 4. and the factorization in line 10. are done using ECM. Hence, in practice one can store the factors found during the smoothness test.

Algorithm 5.1 Index Calculus: main phase

Input: a prime p, a prime divisor π with $\mathrm{val}_\pi(p-1)=1$ and $t \in \mathbb{F}_p^*$ which is not a πth power

Output: $\{\log_t q \bmod \pi \mid q \in \mathcal{F}(B)\}$

1: relations$\leftarrow$empty list
2: **while** #relations $< B$ **do**
3: $h \leftarrow$ random $\big([1, p-2]\big)$
4: **if** $(t^h \bmod p)$ is B-smooth **then**
5: relations$\leftarrow$relations$\bigcup\{h\}$
6: **end if**
7: **end while**
8: M$\leftarrow$empty matrix
9: **for** h in relations **do**
10: Factor $(t^h \bmod p) = \prod_{q\in\mathcal{F}(B)} q^{v(h,q)}$.
11: Add to M the row $[-h, v(h,2), \ldots, v(h,q), \ldots]$ with a coordinate for each $q \in \mathcal{F}(B)$.
12: **end for**
13: Find a non-trivial solution of the $B \times (\#\mathcal{F}(B)+1)$ system $Mu = 0$ over $\mathbb{F}_\pi$
14: Scale u so that the first coordinate is 1
15: Output u

5.2.2 Analysis

Let us explain Algorithm 5.1. In lines 2. $-$ 7. one collects B smooth relations. Consider the relation corresponding to an integer h. When taking logarithms in an arbitrary base t_0 one obtains

$$h \log_{t_0}(t) \equiv \sum_{q\in\mathcal{F}(B)} v(h,q) \log_{t_0}(q) \mod \pi. \tag{5.4}$$

Hence the matrix constructed in lines 8.$-$12. corresponds to a homogeneous system whose unknowns are $\log_{t_0}(q)$ with $q \in \mathcal{F}(B)$ plus an unknown corresponding to $\log_{t_0} t$.

The matrix M has B rows and $\pi(B)+1$ columns. It has therefore approximatively $\log B$ times more rows than columns. A common heuristic is to assume that, with probability $1-o(1)$, the rank of M has its maximal value, i.e., the number of columns minus 1. Then, the system solved in line 13. has a solution space of dimension 1. Each solution corresponds to the discrete logarithms of the elements in the factor base with respect to a generator t_0. The rescaling in line 14. chooses the solution which corresponds to the discrete logarithms in base t.

For the complexity analysis, recall that one can solve a $B\times B$ linear system over a field $\mathbb{F}_\pi$ in $O(B^3)$ field operations using the Gaussian elimination method. This is enough to show that Index Calculus has a sub-exponential complexity. A better complexity is obtained using the observation that the matrix M is sparse. Indeed, each number $(t^h \bmod p)$ is less than p, so $\sum_{q\in\mathcal{F}(B)} v(h,q) \le \log_2 p$. Hence, M has at most $\log_2 p$ non-zero entries per line. The Wiedemann algorithm, presented in

Section 6.3.4, solves the system $Mu = 0$ in $O(B^2\lambda)$ operations, where B is the size of M and λ is the average number of nonzero entries per row. Hence, the linear algebra step in line 13. has a cost of $B^{2+o(1)}$ operations.

Note that the lines $9. - 12.$ have a cost of $B^{1+o(1)}$ operations (see Chapter 2 for the complexity of ECM), being negligible with respect to the linear algebra stage. Finally, the relation collection in lines $2. - 7.$ has a cost of $B/P_{\text{smooth}}(p, B)$ operations. According to Corollary 1.2.1, we obtain

$$\text{time(relation collection)} = L_p\left(1/2, \beta\right) L_p\left(1/2, \frac{1}{2\beta}\right)^{1+o(1)}. \tag{5.5}$$

We optimize the time by equalizing the time of linear algebra and of relation collection, so that we have

$$2\beta = \beta + \frac{1}{2\beta}. \tag{5.6}$$

This implies $\beta = 1/\sqrt{2}$ and the complexity of the main phase is as follows

$$\text{time(main phase)} = L_p\left(1/2, \sqrt{2}\right)^{1+o(1)}. \tag{5.7}$$

5.2.3 Individual logarithm stage

The discrete logarithms obtained by the main phase allow to compute the discrete logarithm of any B-smooth element s. When s is not smooth we use the technique in Algorithm 5.2.

Algorithm 5.2 Index Calculus: individual logarithm

Input: $\{\log_t q \bmod \pi \mid q \in \mathcal{F}(B)\}$ and $s \in \mathbb{F}_p^*$
Output: $\log_t s \bmod \pi$

1: **repeat**
2: h←Random$\big([1, p-2]\big)$
3: **until** $(t^h s \bmod p)$ is B-smooth
4: Factor $(t^h s \bmod p) = \prod_{q\in\mathcal{F}(B)} q^{w(h,q)}$
5: Output $-h + \sum_{q\in\mathcal{F}(B)} w(h, q) \log_t q$.

As the correctness requires no further explanation, we can pass directly to the complexity. The probability of success of each iteration of the main loop equals the probability that a random integer less than p is B-smooth. Hence the average number of loops is $1/P_{\text{smooth}}(p, B)$. With the value of B that we set in the main phase, $B = L_p(1/2, \sqrt{2})$, we obtain

$$\text{time(individual logarithm)} = L_p\left(1/2, \frac{\sqrt{2}}{2}\right)^{1+o(1)}. \tag{5.8}$$

We conclude that the Index Calculus complexity equals that of the main phase:

$$\text{time(Index Calculus)} = L_p\left(1/2, \sqrt{2}\right)^{1+o(1)}. \tag{5.9}$$

5.3 Index Calculus in small characteristic

Index Calculus can be applied to the case of any finite field $\mathbb{F}_{q^k}$, obtaining algorithms of cost $L_{q^k}(1/2, \cdot)$ (see [AD93]). In particular, we show here that it is easy to translate Index Calculus to the case of finite fields of small characteristic.

Indeed, let us consider the DLP in a finite field $\mathbb{F}_{q^k}$ whose elements are represented as elements of $\mathbb{F}_q[X]/\langle\varphi\rangle$ for an irreducible polynomial φ. We call fields of small caracteristic for Index Calculus those for which q is smaller than $L_{q^k}(1/2, 1/\sqrt{2})$. On the one hand, the computations of discrete logarithms in $\mathbb{F}_q^*$ are negligible. On the other hand, since q is small we can finely tune the cardinality of the factor base by changing its maximal degree. The Index Calculus algorithm can be translated with no modification, if one makes φ play the role of p. Indeed, we call x a root of φ in $\mathbb{F}_{q^k}$. The factor base consists in the elements of $\mathbb{F}_{q^k}$ which can be written as polynomials in x of degree less than a constant β:

$$\mathcal{F}(\beta) = \left\{\ell(x) \in \mathbb{F}_q[X] \mid \ell \text{ is monic irreducible and } \deg \ell \leq \beta\right\} \subset \mathbb{F}_{q^k}^*. \quad (5.10)$$

Let t be a polynomial in $\mathbb{F}_q[X]$ such that $t(x)$ generates $(\mathbb{F}_q[X]/\langle\varphi\rangle)^*$. One collects integers h such that the polynomial $t^h \bmod \varphi$ is β-smooth. Then one solves a sparse linear system and obtains the discrete logarithms of elements in $\mathcal{F}(\beta)$. Finally, the individual logarithm stage replicates Algorithm 5.2.

Theorem 1.3.1 shows that the algorithm has a complexity of $L_{q^k}(1/2, \sqrt{2})^{1+o(1)}$, which corresponds to the cost of Index Calculus.

Note though one difference between Index Calculus in prime fields and respectively fields of small characteristic. Let us assume that q is small enough so that one can use Pollard's rho method to compute $\log_t s \bmod (q-1)$. Next, let π be a prime divisor of $\frac{q^n-1}{q-1}$, coprime to $(q-1)$. Let $h \in [1, \pi-1]$ be such that $P(X) := t^h \bmod \varphi(X)$ is β-smooth. We factor $P(X) = a\prod_{i=1}^m p_i(X)$ with $a \in \mathbb{F}_q^*$ and p_i in the factor base (monic). Then we obtain

$$h = \log_t(t^h) \equiv \log_t a + \sum_{i=1}^{m} \log_t(p_i(x)) \bmod \pi. \quad (5.11)$$

Since $a \in \mathbb{F}_q^*$ we have $a^{q-1} = 1$, hence $(q-1)\log_t a \equiv 0 \bmod \pi$, so $\log_t a \equiv 0 \bmod \pi$. This brings us to erase $\log_t a$ from Equation (5.11), so that we can proceed as in the prime case. We tacitly erase logarithms of constant polynomials in all the algorithms of small characteristic.

5.4 The idea of half-relations: illustration by an algorithm of Joux and Lercier

An algorithm of Joux and Lercier [JL06] can be seen as a modification of the polynomial variant of Index Calculus. Although it is a relatively recent method, this algorithm makes a good introduction to modern DLP algorithms. On the one hand, this algorithm uses few mathematical prerequisites; in particular no function fields are necessary. On the other hand, it allows us to explain how the complexity of DLP algorithms was reduced from $L(1/2, \cdot)$ to $L(1/3, \cdot)$.

An alternative would have been to present here the Coppersmith algorithm [Cop84] as the first algorithm of discrete logarithm of complexity $L(1/3, \cdot)$, but it is easier to understand as a particular case of FFS, which uses function fields.

5.4.1 The algorithm

We are given a finite field $\mathbb{F}_{q^k}$ and a prime divisor π of $(q^k-1)/(q-1)$, coprime to $q-1$. We assume that q is less than $L_{q^k}(1/3, c)$ for a constant $c > 0$, so that computing discrete logarithms in $\mathbb{F}_q^*$ is a negligible operation.

Let γ_1 and γ_2 be two polynomials in $\mathbb{F}_q[X]$ such that $\gamma_1(\gamma_2(X)) - X$ has an irreducible factor of degree k. Under the heuristic assumption that this expression behaves as a random polynomial, one can easily find such polynomials by testing random polynomials of degree d_1 and d_2 such that $d_1 d_2 \geq k$. The optimal values of d_1 and d_2 are specified later. Call φ this irreducible factor and represent $\mathbb{F}_{q^k}$ as $\mathbb{F}_q[X]/\langle\varphi\rangle$. Let x be a root of φ in $\mathbb{F}_{q^k}$. Put $y = \gamma_2(x)$ and note that we have

$$\begin{cases} y = \gamma_2(x) \\ x = \gamma_1(y). \end{cases} \tag{5.12}$$

Indeed, $\gamma_1(y) = \gamma_1(\gamma_2(x)) = x$ because the minimal polynomial of x divides $\gamma_1(\gamma_2(X)) - X$. Finally let us define the factor base, for a parameter β to be specified, as in Equation (5.13).

$$\begin{array}{rl} \mathcal{F}(\beta) = & \{\ell(x) \mid \ell \in L_\beta\} \bigcup \{\ell(y) \mid \ell \in L_\beta\} \subset \mathbb{F}_{q^k}^*, \\ \text{where } L_\beta = & \left\{\ell(X) \in \mathbb{F}_q[X] \mid \ell \text{ is monic irreducible and } \deg \ell \leq \beta\right\}. \end{array} \tag{5.13}$$

Contrary to Index Calculus, where one directly searches for equations of the logarithms, we search for pairs of "half-relations". Indeed, for any bivariate polynomial $\phi \in \mathbb{F}_q[X, Y]$ we have

$$\phi\left(x, \gamma_2(x)\right) = \phi(x, y) = \phi\left(\gamma_1(y), y\right). \tag{5.14}$$

We say that a polynomial ϕ of $\mathbb{F}_q[X, Y]$ is doubly-β-smooth if the two univariate polynomials $\phi(X, \gamma_2(X))$ and $\phi(\gamma_1(Y), Y)$ are β-smooth. If ϕ is doubly-β-smooth, we can evaluate in x and y the two corresponding factorizations:

$$\phi(x, \gamma_2(x)) = u_1 \prod_{\ell \in L_\beta} \ell(x)^{v_1(\phi,\ell)} \tag{5.15}$$

$$\phi(\gamma_1(y), y) = u_2 \prod_{\ell \in L_\beta} \ell(y)^{v_2(\phi,\ell)}, \tag{5.16}$$

with u_1 and u_2 in $\mathbb{F}_q^*$. Let t be a generator of $\mathbb{F}_{q^k}^*$. Since $\mathbb{F}_q^*$ has cardinality $q-1$ which is coprime to π, we have $\log_t u_1 \equiv 0 \equiv \log_t u_2 \bmod \pi$. Due to Equation (5.14) we have

$$\sum_{\ell \in L_\beta} v_1(\phi, \ell) \log_t(\ell(x)) \equiv \sum_{\ell \in L_\beta} v_2(\phi, \ell) \log_t(\ell(y)) \mod \pi. \tag{5.17}$$

Algorithm 5.3 The two rational side FFS: main phase

Input: a prime divisor π of $(q^k-1)/(q-1)$ so that $\gcd(\pi, q-1) = 1$; an element t of $\mathbb{F}_{q^k}^*$ whose order is divisible by π, contained in $\mathcal{F}(\beta)$

Output: $\{\log_t z \mod \pi \mid z \in \mathcal{F}(\beta)\}$

1: select γ_1 and γ_2 as above
2: collect $\#\mathcal{F}(\beta)$ pairs (a, b) in $\mathbb{F}_q[X]^2$ with $\deg a, \deg b \leq e, \gcd(a, b) = 1$ and a is monic such that

$$\text{the polynomial } \phi = a(Y) - Xb(Y) \text{ is doubly-}\beta\text{-smooth}$$

3: obtain a matrix M of $\#\mathcal{F}(\beta)$ equations, similar to Equation (5.17)
4: solve the system $Mu = 0$ over $\mathbb{F}_\pi$
5: scale u so that $\log_t t = 1$ and output u

We can now use doubly-β-smooth polynomials, as in Algorithm 5.3; the optimal value of parameter e is specified later.

Note that the condition $t \in \mathcal{F}(B)$ is superfluous. Indeed, one can consider that the main phase of the algorithm computes the discrete logarithms in an arbitrary base. This allows to compute $\log_{t_1} t_2$ for any t_1 and t_2 in the factor base.

5.4.2 Heuristic complexity analysis

Let us choose the values of e and β which minimize the time of Algorithm 5.3. The computations are simplified when we impose that k is at most polynomial in β and e:

$$\log k = O(\log \beta) \text{ and } \log k = O(\log e). \tag{5.18}$$

In order to estimate the time of the linear algebra stage, note that the matrix M is sparse. Indeed, the residues of polynomials $\phi\left(X, \gamma_2(X)\right)$ and $\phi\left(\gamma_1(Y), Y\right)$ modulo φ have degrees at most k, so that each row of M has at most $2k$ non-zero elements. When using the Wiedemann algorithm, presented in Section 6.3.4, one solves the homogeneous system in $O(k\#\mathcal{F}(\beta)^2)$ operations. Since there are q^i monic polynomials of degree i, the number of elements in $\mathcal{F}(\beta)$ is bounded by $2\sum_{i=0}^{\beta} q^i \leq 4q^\beta$. We obtain

$$\text{time(linear algebra)} = kq^{2\beta+o(1)} = q^{2\beta+o(1)}. \tag{5.19}$$

The relation collection stage in line 2. consists in testing the β-smoothness of q^{2e+1} polynomials. On can easily compute [BB07] that the proportion of pairs a, b of degree up to e which are coprime is $1 - \frac{1}{q} + \frac{q-1}{q^{2e}}$. Hence the number of coprime pairs (a, b) is $\Theta(q^{2e})$.

Since the factorization of polynomials over $\mathbb{F}_q$ can be done in probabilistic polynomial time, the relation collection has a total cost of $q^{2e+1}(k \log q)^{O(1)}$ operations:

$$\text{time(relation collection)} = q^{2e+1+o(1)}. \tag{5.20}$$

In order to minimize the overall time we take $e = \beta$. We make the heuristic assumption that polynomials $a(\gamma_2(x)) - xb(\gamma_2(x))$ and $a(y) - \gamma_1(y)b(y)$ have the

same smoothness probability as random polynomials of the same degree. Note that, when no cancellations of the terms occur and a and b have maximal degree, we have

$$\begin{aligned} \deg a(\gamma_2(x)) - xb(\gamma_2(x)) &= d_2 e + 1 \\ \deg a(y) - \gamma_1(y) b(y) &= d_1 + e. \end{aligned} \tag{5.21}$$

The fact that we collect sufficiently many relations translates in

$$q^{2e+1} P_{\text{smooth}}(d_2 e + 1, e) P_{\text{smooth}}(e + d_1, e) \geq 2q^e. \tag{5.22}$$

Since $\gamma_1(\gamma_2(X)) - X$ is required to have a factor of degree k we impose

$$d_1 d_2 \geq k. \tag{5.23}$$

Hence, one minimizes the maximum of the degrees in Equation (5.21) for $d_1 = \sqrt{ke}$ and $d_2 = \sqrt{k/e}$. Joux and Lercier set $e = \frac{1}{\log q} \log\left(L_{q^k}(1/3, \sqrt[3]{4/9})\right)$ and showed that the algorithm has a complexity of $L_{q^k}\left(1/3, \sqrt[3]{32/9}\right)^{1+o(1)}$, which is much faster than the complexity of $L_{q^k}\left(1/2, \sqrt{2}\right)^{1+o(1)}$ for Index Calculus.

Individual logarithm Individual logarithms cannot be computed as in the Index Calculus algorithm. Indeed, already with the smoothness bound of Index Calculus the individual logarithms would take a time $L(1/2, \cdot)$. Moreover, the smoothness bound is smaller in the algorithm of Joux and Lercier. The purpose of this section being to expose the half-relations idea, we skip the details of this stage, which is similar to the algorithm in Section 6.3.5.

Understanding the speed-up This important speed-up can be explained in a simple way. The polynomials which must be β-smooth in Index Calculus have degree k. The degrees in Equations (5.21) are roughly $\sqrt{ke} \approx k^{2/3}$. This allows to take a smoothness bound β equal to roughly $k^{1/3}$ and to have a probability

$$\text{Prob}\left(\text{doubly-}k^{1/3}\text{-smooth}\right) = \rho\left(k^{1/3}\right)^2 = L(1/3, \cdot), \tag{5.24}$$

which is to be compared to $\rho(\sqrt{k}\sqrt{\log k}/2) = L(1/2, \cdot)$ in the polynomial variant of Index Calculus. In short, one replaced the probability that a polynomial of large degree is smooth by the probability that two polynomials of very small degree are simultaneously smooth.

Chapter 6

Overview on NFS and FFS

In the previous chapter we have postponed as much as possible the technicalities inherent to discrete logarithm algorithms of complexity $L(1/3)$. Now we enter the core of the difficulties and provide an overview of FFS and NFS for discrete logarithms in finite fields. We do not spend too much time on the basics of number and function fields as there exist good references on the topic. On the contrary we give a relatively long description of NFS at the highest level of presentation. It will allow us to make a complexity analysis and to propose an improvement in Chapter 7. We leave some of the classical improvements for Chapter 8 because they are cumbersome at the first contact with the algorithm. We will also focus on virtual logarithms and on the decomposition of problematic primes, topics which can be avoided in the factoring algorithms.
We organize the chapter as follows. After a short recapitulation on the mathematical background, we present NFS and virtual logarithms. Then we introduce FFS, first by analogy with NFS, then with explicit details on each step.

6.1 Prerequisites on function and number fields

Let us review the mathematical foundations of the two most important sieving algorithms: NFS and FFS. Our presentation follows the description in [Neu99] and [Lor96], which can be consulted for reference. Given an irreducible polynomial f in $\mathbb{F}_q(t)[x]$, we call function field of f the field $\mathbb{F}_q(t)[x]/\langle f\rangle$. Similarly, any number field, i.e., finite degree extension of $\mathbb{Q}$, can be written as $\mathbb{Q}[x]/\langle f\rangle$ for an irreducible polynomial f in $\mathbb{Q}[x]$. The elements of a number field are called algebraic numbers whereas the elements of a function field are called algebraic functions.

The common properties of the fields of rational numbers $\mathbb{Q}$ and of rational functions $\mathbb{F}_q(t)$ imply a series of similarities between the number and function fields. This is why we list their properties in a unified manner.

Theorem 6.1.1. *Let K be a number field (resp. function field). Let $\mathcal{O}_K$ be the set of elements of K whose minimal (monic) polynomial has coefficients in $\mathbb{Z}$ (resp.*

$\mathbb{F}_q[t]$). *Then* $\mathcal{O}_K$ *is a Dedekind ring, in particular every ideal of* $\mathcal{O}_K$ *factors into prime ideals in a unique way up to the order of the factors.*

A key element of the effectiveness of this theorem is the factorization of the elements $\ell\mathcal{O}_K$ for primes (resp. irreducible polynomials) ℓ. In the sequel f is supposed to be monic with coefficients in $\mathbb{Z}$ (resp. $\mathbb{F}_q[t]$); the case of non-monic polynomials will be studied in Section 6.5. Let us call conductor of f the smallest positive integer (resp. monic polynomial) c such that $c\mathcal{O}_K \subset \mathbb{Z}[\alpha]$ (resp. $c\mathcal{O}_K \subset \mathbb{F}_q[t][\alpha]$), where α is a root of f in K.

Proposition 6.1.2. *Let* K *be a number (resp. function) field defined by the monic polynomial* f *with coefficients in* $\mathbb{Z}$ *(resp.* $\mathbb{F}_q[t]$*). Let* ℓ *be a prime (resp. irreducible polynomial) coprime to the conductor* c *of* f*. Then the factorization of the ideal* $\ell\mathcal{O}_K$ *is given by*

$$\ell\mathcal{O}_K = \prod_{i=1}^{r} \langle \ell, f_i(\alpha)\rangle^{e_i}, \tag{6.1}$$

where

$$(f(X) \bmod \ell) = \prod_{i=1}^{r} f_i(X)^{e_i} \tag{6.2}$$

is the factorization of f *in* $\mathbb{F}_\ell[X]$ *(resp.* $(\mathbb{F}_q[t]/\langle \ell\rangle)[X]$*).*

If K/Q is a field extension of finite degree, for any $\gamma \in K$ we call χ_γ its characteristic polynomial over Q and call norm of γ over Q the constant term of χ_γ; it is denoted $\mathrm{N}_{K/Q}(\gamma)$ or $\mathrm{N}(\gamma)$ when K and Q are clear from the context. The norm of an algebraic number (resp. function) plays a key role in NFS and FFS.

Proposition 6.1.3. *Let* K *be a number (resp. function) field of defining polynomial* $f \in \mathbb{Z}[x]$ *(resp.* $f \in \mathbb{F}_q[t][x]$*) and* α *a root of* f *in* K*. Assume that* f *is monic.*[1] *Then, for any* a *and* b *in* $\mathbb{Q}$ *(resp.* $\mathbb{F}_q(t)$*) we have*

$$\mathrm{N}(a - \alpha b) = F(a, b), \tag{6.3}$$

where $F(x, y) = f(x/y)y^{\deg f}$ *is the homogenization of* f*.*

One also defines the norm of an ideal $\mathfrak{u}$ as $N(\mathfrak{u}) := [\mathcal{O}_K : \mathfrak{u}]$. The most important property is multiplicativity: for all ideals I and J we have $\mathrm{N}(IJ) = \mathrm{N}(I)\,\mathrm{N}(J)$. Also, if $\gamma \in \mathcal{O}_K$ then one has $\mathrm{N}(\gamma) = \pm\,\mathrm{N}(\gamma\mathcal{O}_K)$. Let us see how to effectively factor an ideal of type $(a - b\alpha)\mathcal{O}_K$ into prime ideals.

Proposition 6.1.4 (Dedekind's method)**.** *Let* K *be a number field defined by a polynomial* f *with coefficients in* $\mathbb{Z}$ *(resp.* $\mathbb{F}_q[t]$*). Assume that* f *is monic and call* α *a root of* f *in* K *and* c *the conductor of* f*. Let* a *and* b *be two elements of* $\mathbb{Z}$ *(resp.* $\mathbb{F}_q[t]$*) such that* $\gcd(a, b) = 1$ *and* $\gcd(\mathrm{N}(a - b\alpha), c) = 1$*. Then one has*

$$(a - b\alpha)\mathcal{O}_K = \prod_{\ell | F(a,b)} \langle \ell, \alpha - ab^{-1} \bmod \ell\rangle^{\mathrm{val}_\ell F(a,b)}. \tag{6.4}$$

[1] If α is root of a polynomial over $\mathbb{Z}$ whose leading coefficient f_d is not 1, then one obtains $F(a, b) = f_d\,\mathrm{N}(a - b\alpha)$.

Proof. If I is an ideal which divides $(a-b\alpha)\mathcal{O}_K$, then $\mathrm{N}(I)$ divides $\mathrm{N}(a-b\alpha)$. Hence all the divisors of $(a-b\alpha)\mathcal{O}_K$ are above primes (resp. irreducible polynomials) which divide $\mathrm{N}(a-b\alpha)$.

Let ℓ be a prime divisor of $\mathrm{N}(a-b\alpha) = F(a,b)$. By Proposition 6.1.2, as $\gcd(c,\ell)=1$, we have that the ideals of $\mathcal{O}_K$ above ℓ are given by $\langle \ell, P(\alpha)\rangle$ where P are irreducible factors of $\overline{f}$, the reduction of f modulo ℓ.

Since $\gcd(a,b)=1$, $bX-a \bmod \ell$ is irreducible and it is the unique divisor of $a-bX \bmod \ell$ up to a scalar factor. Therefore the ideal $\mathfrak{l} := \langle \ell, b\alpha - a \bmod \ell\rangle$ is prime and it is the unique divisor of $(a-b\alpha)\mathcal{O}_K$ above ℓ. One can check that $[\mathcal{O}_K : \mathfrak{l}] = \ell$. Hence, the exponent of $\mathfrak{l}$ in $(a-b\alpha)\mathcal{O}_K$ is the valuation of ℓ in $\mathrm{N}((a-b\alpha)\mathcal{O}_K) = \pm\mathrm{N}(a-b\alpha)$. Finally, since f is monic and $\gcd(a,b)=1$ we have $b \not\equiv 0 \bmod \ell$. Therefore $\mathfrak{l} = \langle \ell, \alpha - ab^{-1} \bmod \ell\rangle$ and we obtain the result. □

Since finding a basis of $\mathcal{O}_K$ is a computationally hard problem [Coh93], we cannot evaluate the conductor c and therefore it is common to replace c in the statement above by the discriminant $\mathrm{Disc}(f)$ which is a multiple of c. The condition that f is monic can be replaced by the condition that ℓ is coprime to the leading coefficient of f. One has therefore an algorithm to factor $(a-b\alpha)\mathcal{O}_K$ for all coprime pairs (a,b) for which the norm is coprime to $\mathrm{Disc}(f)$ and the leading coefficient of f. This is enough for a theoretical study of NFS and FFS. In practice, one prefers to use all the coprime pairs (a,b); hence a general method of factoring $(a-b\alpha)\mathcal{O}_K$ will be presented in Section 6.5.

The following object is computationally hard and is used exclusively in the analysis of the algorithms. We call fractional ideal any subset of K of the form $\frac{1}{m}I$ where I is a non-zero ideal of $\mathcal{O}_K$ and a m is a non-zero element of $\mathbb{Z}$ (resp. $\mathbb{F}_q[t]$)). It is known that the fractional ideals form a group [Neu99]. An ideal of type $\gamma\mathcal{O}_K$ with $\gamma \in K$ is called principal ideal.

Proposition 6.1.5. *Let K be a number (resp. function) field. We call G the group of fractional ideals and P the group of principal ideals. Then the order of the quotient group G/P, called class group, is finite. Its order is called the class number of K.*

The group $\mathcal{O}_K^*$ of invertible elements in $\mathcal{O}_K$ is abelian and its rank can easily be computed as in the following result.

Theorem 6.1.6. *Let K be a number field defined by a polynomial f in $\mathbb{Q}[x]$. We denote by $n_{\mathbb{R}}$ and $n_{\mathbb{C}}$ the number of real roots of f and half of the number of roots in $\mathbb{C}\backslash\mathbb{R}$. Then we have*

$$\mathcal{O}_K^* = U \times \mathbb{Z}^{n_{\mathbb{R}}+n_{\mathbb{C}}-1}, \tag{6.5}$$

where U is the group of roots of unity contained in K.

The pair $(n_{\mathbb{R}}, n_{\mathbb{C}})$ is called the signature of K and one usually denotes $n_{\mathbb{R}}+n_{\mathbb{C}}-1$ by r_K. Any basis of the non-torsion part of $\mathcal{O}_K^*$ is called system of fundamental units.

In the number field sieve, we will use a simple criterion to guarantee that a number field K has no roots of unity of a given order.

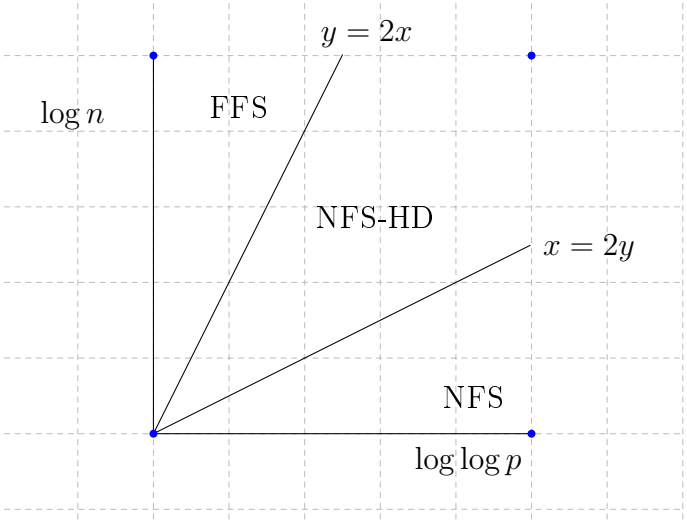


Figure 6.1: The domain of DLP algorithms in $\mathbb{F}_Q^*$ with $Q = p^n$.

Lemma 6.1.7. *Let K be a number field and π an odd prime. If π does not divide* $\mathrm{Disc}(K)$*, then K has no πth root of unity.*

Proof. Suppose that K has a root ζ_π of order π. Then, K contains $\mathbb{Q}(\zeta_\pi)$ and, in particular $\mathrm{Disc}(\mathbb{Q}(\zeta_\pi))$ divides $\mathrm{Disc}(K)$. Since $\mathrm{Disc}(\zeta_\pi)$ is a power of π [Neu99], $\mathrm{Disc}(K)$ must be divisible by π. □

The case of function fields is at least as easy as that of number fields. Indeed, Equation (6.5) translates to the case of function fields when one replaces U with $\mathbb{F}_q^*$ (the field of constants) and $n_{\mathbb{R}} + n_{\mathbb{C}}$ by the number of places at infinity. We will return to this matter in Section 6.6.3.

6.2 The Number Field Sieve

The Number Field Sieve (NFS) [Gor93],[JL03] and the Function Field Sieve (FFS) [Adl94],[JL02] are the most studied algorithms dedicated to the DLP in finite fields. While NFS applies to fields of large characteristic, in particular to fields $\mathbb{Z}/p\mathbb{Z}$ with p prime, FFS applies to fields $\mathbb{F}_{q^k}$ with small q. A variant of NFS, the Number Field Sieve in High Degree (NFS-HD) [JLSV06], applies to the intermediate cases so that one can solve the DLP in any finite field $\mathbb{F}_Q$ in time $L_Q(1/3, c)$ where c is an explicit constant. Figure 6.1 shows the domain of application for NFS, NFS-HD and FFS respectively.

Similarly to Index Calculus, in NFS and FFS we define a factor base, we collect relations among the elements of the factor base, we solve a linear system to compute discrete logarithms associated to the factor base and then we deduce the requested logarithm. We also use the idea of "half-relations" that we presented in Section 5.4, hence using two number (resp. function) fields. Although the description in this section applies to the three algorithms–NFS, FFS and NFS-HD–, for simplicity we focus on NFS, and we consider the particular case of prime fields.

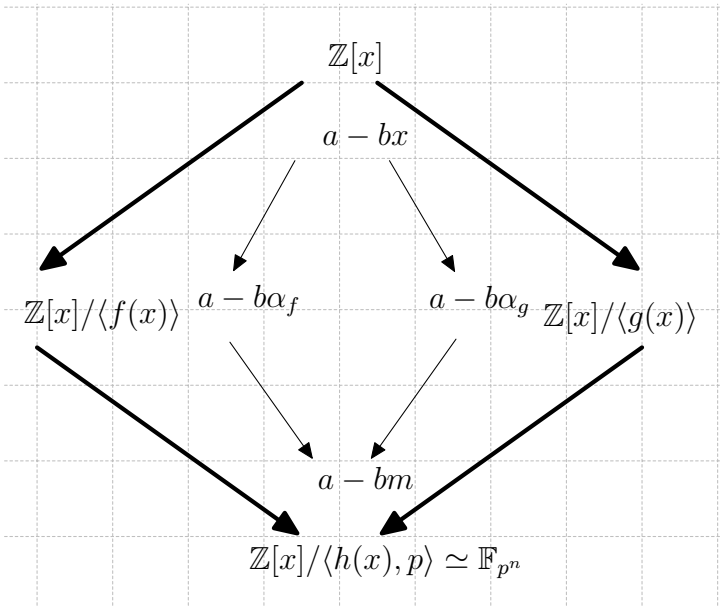


Figure 6.2: The NFS commutative diagram.

Let $\mathbb{F}_p$ be a prime field in which one wants to solve the DLP. Let f_1 and f_2 be two irreducible polynomials in $\mathbb{Z}[x]$ whose reductions $\overline{f_1}$ and $\overline{f_2}$ modulo p have a common root m. For simplicity we further assume that f_1 and f_2 are monic, postponing the general case until Section 6.5. Call α_1 and α_2 two roots of f_1 and f_2 in their respective number fields K_1 and K_2. As f_1 and f_2 are monic we know that α_1 and α_2 are integers in K_1 and K_2.

In what follows, any object of index i refers to f_i for $i = 1, 2$. The main property of f_1 and f_2 is that there are two ring homomorphisms from $\mathbb{Z}[\alpha_1]$ and $\mathbb{Z}[\alpha_2]$ to $\mathbb{F}_p$:

$$\begin{array}{rcl} \mathbb{Z}[\alpha_i] & \to & \mathbb{F}_p \\ P(\alpha_i) & \mapsto & P(m), \end{array} \tag{6.6}$$

where P is any polynomial in $\mathbb{Z}[x]$. Note that, for all $P \in \mathbb{Z}[x]$, $P(\alpha_1)$ and $P(\alpha_2)$ are mapped to the same element $P(m)$, as illustrated by the diagram in Figure 6.2.

Let B be an integer called the smoothness bound. The factor base of f_i, denoted $\mathcal{F}_i(B)$, is the set of prime ideals of the form $\langle \ell, \alpha_i - r\rangle$ with ℓ a prime below B and r such that $f_i(r) \equiv 0 \bmod \ell$. Note that for the primes ℓ not dividing $\mathrm{Disc}(f_i)$ nor the leading coefficient of f_i these are exactly the degree 1 prime ideals of $\mathcal{O}_i$ of norm less than B. The factor base of the algorithm is $\mathcal{F}(B) := \mathcal{F}_1(B) \bigcup \mathcal{F}_2(B)$.

We say that a pair $(a, b) \in \mathbb{Z}^2$ is doubly-B-smooth if $\mathrm{N}(a - b\alpha_1)$ and $\mathrm{N}(a - b\alpha_2)$ are both B-smooth. For ease of presentation, in this section we forget the primes which divide $\mathrm{Disc}(f_1)\,\mathrm{Disc}(f_2)$ and the leading coefficients of the two polynomials. Indeed, they require special algorithms and are analyzed in Section 6.5. Hence, in this section a pair is doubly-B-smooth if, in addition to the previous condition, the two norms are coprime to the discriminants and the leading coefficients of f_1 and f_2.

Let (a, b) be a doubly-B-smooth pair such that $\gcd(a, b) = 1$. We can write

$$(a - b\alpha_1)\mathcal{O}_1 = \prod_{\mathfrak{l} \in \mathcal{F}_1(B)} \mathfrak{l}^{v_1(a,b,\mathfrak{l})} \tag{6.7}$$

$$(a - b\alpha_2)\mathcal{O}_2 = \prod_{\mathfrak{l} \in \mathcal{F}_2(B)} \mathfrak{l}^{v_2(a,b,\mathfrak{l})}. \tag{6.8}$$

Let us assume that, when K is a number or function field, the following problems are computationally easy:

1. computing the class number h of K;
2. finding a generator for any principal ideal P of $\mathcal{O}_K$;
3. determining a system of fundamental units $u_1, \ldots, u_r$ of $\mathcal{O}_K^*$;

One can check that, given a basis $\{u_j \mid j \in [1, r]\}$ of $\mathcal{O}_K^*$ and a unit u, one can easily compute a set of exponents e_j with $j \in [1, r]$ and a root of unity κ such that $u = \kappa \prod_{j=1}^{r} u_j^{e_j}$.

These hard-to-meet assumptions are satisfied for the number fields of polynomials f with very small coefficients. For example, one of the polynomials used by Joux and Lercier in [JL03] was $f = x^3+2$. For its number field one has $\mathcal{O} = \mathbb{Z}[\sqrt[3]{2}]$ and all prime ideals are principal. If $\mathfrak{l}$ is a degree 1 ideal $\langle \ell, -\sqrt[3]{2}-r \rangle$, then $\mathfrak{l} = (a_\mathfrak{l}+b_\mathfrak{l}\sqrt[3]{2})\mathcal{O}$ where $a_\mathfrak{l}$ and $b_\mathfrak{l}$ are two integers such that $a_\mathfrak{l}+b_\mathfrak{l}\sqrt[3]{2} = \gcd(\ell, -\sqrt[3]{2}-r)$ in the principal (but not Euclidean) ring $\mathbb{Z}[\sqrt[3]{2}]$. Finally, $u_1 := 1 - \sqrt[3]{2}$ is a fundamental unit and any unit can be written as $\pm u_1^n$ for an integer n.

For primes of less than 100 decimal digits the best value of d seems to be 2. In this case one can privilege the polynomials $f = x^2 + r$ with $r \in \{1, 2, 3, 7, 11, 19, 43, 67, 163\}$. Indeed, for these values of r the ring $\mathcal{O}$ is principal and the unit rank is 0. We obtain the Gaussian integer algorithm [COS86],[Web97] as a particular case of NFS.

Returning to the general case, for $i = 1, 2$ and for each ideal $\mathfrak{l}$ in $\mathcal{F}_i(B)$ we know that $\mathfrak{l}^{h_i}$ is principal where h_i is the class number of K_i. Under the assumption 2 we can find an element $\gamma_\mathfrak{l}$ such that, for $i = 1, 2$,

$$\mathfrak{l}^{h_i} = \gamma_\mathfrak{l}\mathcal{O}_i. \tag{6.9}$$

Hence Equations (6.7) and (6.8) become

$$(a - b\alpha_1)^{h_1} = U_1 \prod_{\mathfrak{l} \in \mathcal{F}_1(B)} \gamma_\mathfrak{l}^{v_1(a,b,\mathfrak{l})} \tag{6.10}$$

$$(a - b\alpha_2)^{h_2} = U_2 \prod_{\mathfrak{l} \in \mathcal{F}_2(B)} \gamma_\mathfrak{l}^{v_2(a,b,\mathfrak{l})}, \tag{6.11}$$

for some units $U_1 \in \mathcal{O}_1$ and $U_2 \in \mathcal{O}_2$. Under assumption 3 we can effectively write $U_1 = \kappa_1 \prod_{j=1}^{r_1} u_{1,j}^{e_1(a,b,j)}$ and similarly for U_2. Being elements of $\mathbb{Q}(\alpha_1)$ and $\mathbb{Q}(\alpha_2)$, the units U_1, U_2 and the elements $\gamma_\mathfrak{l}$ can be written as $P(\alpha_1)$ or $P(\alpha_2)$ for some polynomials P in $\mathbb{Q}[x]$.

We know how to map in $\mathbb{F}_p$ any element of $\mathbb{Z}[\alpha_1]$ or $\mathbb{Z}[\alpha_2]$ due to Equation (6.6) or to Figure 6.2. One can extend the morphism to elements γ in K whose norm is coprime to p. Indeed, let us put $\mathfrak{p}_1 = \langle p, \alpha_1 - m\rangle$ and $\mathfrak{p}_2 = \langle p, \alpha_2 - m\rangle$. For any $\gamma \in \mathcal{O}_1$ (resp $\mathcal{O}_2$), whose norm denominator is coprime to p, we denote by $\overline{\gamma}$ the reduction of γ modulo $\mathfrak{p}_1$ (resp. $\mathfrak{p}_2$): if $\gamma = P(\alpha_1)/Q$ (resp. $P(\alpha_2)/Q$) for some $P \in \mathbb{Z}[X]$ and $Q \in \mathbb{Z}$ with Q coprime to p, then $\overline{\gamma} = \overline{P}(m)\overline{Q}^{-1}$ where $\overline{P}$ and $\overline{Q}$ are the reductions of P and Q modulo p.

We finally obtain

$$\left(\overline{\kappa_1}\prod_{i=1}^{r_1}\overline{u_{1,i}}^{e_1(a,b,i)}\prod_{\mathfrak{l}\in\mathcal{F}_1(B)}\overline{\gamma_\mathfrak{l}}^{v_1(a,b,\mathfrak{l})}\right)^{h_1^{-1} \bmod \pi} = \left(\overline{\kappa_2}\prod_{i=1}^{r_2}\overline{u_{2,i}}^{e_2(a,b,i)}\prod_{\mathfrak{l}\in\mathcal{F}_2(B)}\overline{\gamma_\mathfrak{l}}^{v_2(a,b,\mathfrak{l})}\right)^{h_2^{-1} \bmod \pi} \tag{6.12}$$

This equation involves only elements in $\mathbb{F}_p$. We make the additional assumption that the class numbers h_1 and h_2 are coprime to the large prime factor π of $p-1$ which is the modulus of our logarithms. Since discrete logarithms can be computed with Pollard's method in time $L_p(1/3,\cdot)$ when $\pi < L_p(1/3,c)$ for a constant $c > 0$, we can assume $\pi > L_p(1/3,1)$, so that the probability that it divides h_1 or h_2 is tiny.

For $i = 1, 2$, the order of $\overline{\kappa_i}$ in $\mathbb{F}_p^*$ divides the order of κ_i in $\mathcal{O}_i$. We assume that π is coprime to $\mathrm{Disc}(f_i)$, which is highly probable because $\pi > L_p(1/3,1)$. By Lemma 6.1.7, K_1 and K_2 have no πth root of unity. Then, for any generator t of $\mathbb{F}_p^*$ we have

$$\log_t \overline{\kappa_1} \equiv \log_t \overline{\kappa_2} \equiv 0 \mod \pi. \tag{6.13}$$

We can now write an additive equation in terms of the logarithms $\log_t(u_{1,i})$, $\log_t(u_{2,i})$ and $\log_t \gamma_\mathfrak{l}$ for $\mathfrak{l}$ in $\mathcal{F}(B)$:

$$\begin{aligned}\textstyle\sum_{i=1}^{r_1} e_1(a,b,i)\frac{\log_t(\overline{u_{1,i}})}{h_1} + \sum_{\mathfrak{l}\in\mathcal{F}_1(B)} v_1(a,b,\mathfrak{l})\frac{\log_t(\overline{\gamma}_\mathfrak{l})}{h_1} \equiv \\ \textstyle\sum_{i=1}^{r_2} e_2(a,b,i)\frac{\log_t(\overline{u_{2,i}})}{h_2} + \sum_{\mathfrak{l}\in\mathcal{F}_2(B)} v_2(a,b,\mathfrak{l})\frac{\log_t(\overline{\gamma}_\mathfrak{l})}{h_2} \bmod \pi.\end{aligned} \tag{6.14}$$

The quantities $h_1^{-1}\log_t\gamma_\mathfrak{l}$ and $h_2^{-1}\log_t\gamma_\mathfrak{l}$ are called virtual logarithms of $\mathfrak{l}$ and will be analyzed in detail in Chapter 8. We have shown how to obtain an equation among the virtual logarithms for any doubly-B-smooth and coprime pair (a,b). We make the additional heuristic that the linear system obtained when writing equations in this manner has a kernel of dimension 1. Then one can compute the virtual logarithms of all elements in the factor base.

We can now present the main phase of the NFS algorithm, for which B and E are parameters that will be discussed later.

6.3 Detailed presentation of NFS stages

Algorithm 6.1 can be divided into stages as follows: Polynomial selection (line 1), Relation collection (line 2), Filtering (line 3 and the beginning of 4) and Linear algebra stage (line 4). As in the case of Index Calculus, a second algorithm called Individual logarithm is used to compute the discrete logarithm of any given element s of $\mathbb{F}_p^*$. In this section we limit ourselves to a short presentation, but we will return to the Relation collection and the Linear algebra stages in Chapter 8.

Algorithm 6.1 The Number Field Sieve: main phase

Input: a prime p, a prime factor π of $p-1$, an element $t \in \mathbb{F}_p^* \bigcap \mathcal{F}(B)$ of order multiple of π

Output: virtual logarithms of ideals $\mathfrak{l}$ in $\mathcal{F}(B)$ and of the fundamental units of K_1 and K_2

1: Select two irreducible polynomials f_1 and f_2 with a common root m modulo p
2: Collect $\#\mathcal{F}(B)$ coprime pairs $(a,b) \in \mathbb{Z}^2$ with $|a|, |b| \leq E$ and
$$\mathrm{N}(a - b\alpha_1) \text{ and } \mathrm{N}(a - b\alpha_2) \text{ are } B\text{-smooth}$$
3: Construct a matrix M from the additive equations
4: Solve $Mu = 0$ over $\mathbb{F}_\pi$
5: Scale u so that $\log_t t = 1$ and output u

6.3.1 Polynomial selection

In order to keep the classical notations, from now on, we will call f and g the two polynomials f_1 and f_2 used in NFS. Recall that NFS requires that f and g are irreducible in $\mathbb{Z}[x]$ and that the reductions $\overline{f}$ and $\overline{g}$ modulo p have a common root m. Before seeing more complex methods in the next chapter, let us see the so-called base-m method in Algorithm 6.2.

Algorithm 6.2 Polynomial selection: the base-m method

Input: a prime p and an integer parameter d

Output: two polynomials f and g, with $\deg f = d$ and $\deg g = 1$, sharing a root m modulo p

1: $m \leftarrow \left\lceil \sqrt[d]{p} \right\rceil$
2: **repeat**
3: $\quad m \leftarrow m - 1$
4: $\quad$ Write p in base m: $p = \sum_{i=0}^{d} p_i m^i$
5: $\quad f \leftarrow \sum_{i=0}^{d} p_i x^i$ and $g \leftarrow x - m$
6: **until** f is irreducible
7: Output f and g

It is important to note that f is monic and we have

$$\max\left(\|f\|_\infty, \|g\|_\infty\right) \leq m \leq p^{1/d}. \tag{6.15}$$

Indeed, note first that the coefficients of f and g are bounded by m which is smaller than $p^{1/d}$. Next, since a random polynomial in $\mathbb{Z}[x]$ is heuristically always irreducible, the loop in Algorithm 6.2 has a small number of iterations, so that $m > (p/2)^{1/d}$. This proves that f is monic.

6.3.2 Relation collection: the sieve

Recall that, given a parameter E, NFS collects all the coprime pairs (a,b) with $|a|, |b| \leq E$ such that the homogenizations of the two polynomials evaluated at

(a, b), $F(a, b)$ and $G(a, b)$, are B-smooth for a parameter B. Since pairs (a, b) and $(-a, -b)$ give the same additive relation, we restrict to the pairs (a, b) in the rectangle $[0, E] \times [-E, E]$.

The naive approach would be to enumerate all the pairs (a, b) and to test the B-smoothness of $F(a, b) \cdot G(a, b)$ using ECM. As we show in Chapter 7, the complexity of relation collection stage is $L_p(1/3, \cdot)^{1+o(1)}$ and the smoothness bound $B = L_p(1/3, \beta)$ for some constant $\beta > 0$. According to Proposition 1.2.2 the time of each ECM test with smoothness bound B is $L_p(1/6, \cdot)$. Hence the difference between this naive technique and any other technique which enumerates the sieving domain $[0, E] \times [-E, E]$ is hidden in the $o(1)$ term in the expression of the complexity. Nevertheless, it is important to gain a factor $L_p(1/6, \cdot)$ on an algorithm of complexity $L_p(1/3, \cdot)$. For this we use a "sieving" procedure inspired from Erathostene's sieve.

The basic idea is that if $F(x, y)$ is a bivariate polynomial and ℓ a prime, then one can easily mark all the pairs (a, b) such that $F(a, b) \equiv 0 \bmod \ell$. Now suppose that each pair (a, b) in the sieving rectangle $[0, E] \times [-E, E]$ received a mark for each prime ℓ which divides $F(a, b)$. Then the values of $F(a, b)$ which are prime are exactly these which have no mark. In the opposite case, if we put marks only for primes ℓ less than the smoothness bound B, then the pairs (a, b) which have many marks have a higher probability that $F(a, b)$ is B-smooth.

In more detail, suppose that we know all the pairs (ℓ^k, r) such that $f(r) \equiv 0 \bmod \ell^k$. Let ℓ be a prime and $k \geq 1$ an integer. Since f is monic, the pairs (a_0, b_0) in $[0, \ell^k]^2$ such that $\gcd(a, b, \ell) = 1$ and $F(a_0, b_0) \equiv 0 \bmod \ell^k$ are exactly those such that $a \equiv br \bmod \ell^k$. Hence, we can easily mark all the pairs (a, b) with $a \equiv a_0 \bmod \ell^k$ and $b \equiv b_0 \bmod \ell^k$.

The sieving procedure goes as follows. We make an array whose entries are indexed by pairs (a, b) and we initialize it to $\log_2 F(a, b)$. Then, for each pair (ℓ, k) we subtract $\log_2 \ell$ from array entries indexed by pairs (a, b) such that $F(a, b) \equiv 0 \bmod \ell^k$. Note that in total, for each pair (a, b) we subtract $\mathrm{val}_\ell(F(a, b))$ times $\log_2 \ell$ from the entry of index (a, b). The pairs such that $\log_2 F(a, b) = \sum_{\ell \in \mathcal{F}(a,b)} \log_2 \ell \, \mathrm{val}_\ell F(a, b)$ are exactly the smooth pairs. In practice, we work with approximations of logarithms, so that we keep for a smoothness test all the pairs which after the sieve have a value less than a threshold parameter. We give a precise description in Algorithm 6.3. We run it for f and then for g and we obtain the doubly-B-smooth pairs by comparing the output of the two sieves.

The reader might object to the description of Algorithm 6.3 as the folklore contains many modifications. Note first that NFS does not require all the doubly-smooth pairs (a, b), but rather it requires that the number of relations exceeds the cardinality of the factor base. Hence, one prefers to lose some of the relations if one obtains a good ratio relations/second. Additionally, the smoothness test used in practice is ECM, which is a probabilistic algorithm. Finally, it is not clear if one must keep the pairs (ℓ^k, r) in line 1. for $k \geq 2$ and $\ell \geq 5$. Indeed, sieving the powers is costly and it might not reduce by much the number of pairs which pass the smoothness test. For example the earlier versions of the CADO-NFS software used to skip sieving for the powers.

Note that from a theoretical point of view the parameter τ can be set to 0

Algorithm 6.3 Line sieving

Input: a monic polynomial $f(x)$ in $\mathbb{Z}[x]$
a parameter $\tau > 0$ (threshold)

Output: all the coprime pairs $(a, b) \in [0, E] \times [-E, E]$ such that $F(a, b)$ is B-smooth

Make a list (ℓ^k, r) of prime powers $\ell \leq B$ and integers $0 \leq r < \ell$ such that $f(r) \equiv 0 \bmod \ell^k$
Define an array indexed by the pairs (a, b) in $[0, E] \times [-E, E]$ and initialize it with $\log_2 F(a, b)$
for all (ℓ^k, r) **do**
    for b in $[-E, E]$ **do**
        $a_0 \leftarrow br \bmod \ell^k$
        for u in $\left[-\lfloor \frac{a_0}{\ell^k} \rfloor, \lfloor \frac{E-a_0}{\ell^k} \rfloor\right]$ **do**
            Subtract $\log_2 \ell$ from the entry of indices $(a_0 + u\ell^k, b)$
        end for
    end for
end for
for pairs (a, b) in $[0, E] \times [-E, E]$ **do**
    if $\gcd(a, b) = 1$ and the value in the entry of (a, b) is less than τ **then**
        if a B-smoothness test on $F(a, b)$ with ECM answers true **then**
            Output (a, b)
        end if
    end if
end for

so that the ECM test can be skipped. From a practical point of view this test is taken by a small fraction of pairs, so that the actual sieving in lines $3. - 10.$ and the "cofactorization" in lines $11. - 17.$ take a relatively equal amount of time. Note that the cofactorization gives us the factors of the norms, which are needed for the next stage of the algorithm.

6.3.3 Filtering

Sometimes seen as part of the linear algebra stage of NFS, the filtering transforms the set of pairs (a, b) into a matrix from which it eliminates the non-necessary rows and columns. It is identical for NFS[2], FFS and NFS-HD and has strong similarities in the factorization algorithms.

When writing an equation for a pair (a, b), one starts by factoring $a - b\alpha$ and $a - bm$ into degree 1 ideals as in Proposition 6.1.4. Next, one computes coefficients corresponding to the units of $\mathcal{O}_f$ using the Schirokauer maps, as explained in Section 6.4.

One can finally simplify the matrix as follows. If one unknown occurs in a unique equation, called singleton, one stores the equation for later use and deletes the row and the column from the matrix. Secondly, if one unknown occurs in exactly two equations, then we can subtract one from the other. This transforms one of the two equations into a singleton that can be erased. If the two equations have w_1 respectively w_2 non-zero entries, then the combined equation has at most $w_1 + w_2 - 2$ non-zero entries. Hence, the weight, i.e., the number of non-zero entries, of the matrix is decreased as well as its number of rows and columns. This stage is called clique-removal and is followed by a stage, called merge, corresponding to the first operations of the Gaussian elimination algorithm. A thorough analysis of the Filtering can be found in [LO91],[PS92],[Cav00], and [Bou13].

6.3.4 The linear algebra stage

In short, the linear algebra stage takes as input a square matrix M of zero determinant with coefficients over $\mathbb{F}_\pi$ for a large prime π and computes a non-trivial vector u such that $Mu = 0$. The stage occurs in various discrete logarithm algorithms, starting with the sieving algorithms presented in this chapter, continuing with other algorithms of finite fields [AM12],[Jou13b] and ending with algorithms for algebraic curves [Gau09] and [JV12].

Note first that the matrix M obtained in NFS (resp. FFS) is sparse. Indeed, apart from up to $\deg f + \deg g$ columns of the matrix, called the Schirokauer maps, that we introduce in Section 6.4, the non-zero entries of any row (a, b) are the exponents of the prime divisors of $F(a, b)G(a, b)$ (see Proposition 6.1.4). Since any number less than p has at most $\log_2 p$ prime divisors and by the analysis in Section 7.1 $|F(a, b)|$ and $|a - bm|$ are less than d, each row of the matrix has at most $2 \log_2 p + \deg f + \deg g$ non-zero entries.

[2] In this document NFS stands for the DLP algorithm; the corresponding factorization algorithm having a different filtering stage.

Two of the fastest algorithms for solving sparse homogeneous systems are Wiedemann's and Lanczos' algorithms. Their complexities are $O(N^2\lambda)$ where λ is the average number of non-zero entries per row and N the number of rows and columns of M.

In this document we focus on the former because it has better parallelism properties, as we argue in Section 8.2.2. The basic idea is to consider any polynomial $C \in \mathbb{F}_\pi[x]$, $C(x) = \sum_{i=0}^{\deg C} c_i x^i$ such that $C(M) = 0$:

$$\sum_{i=0}^{\deg C} c_i M^i = 0. \tag{6.16}$$

Since M is singular, i.e., $\det M = 0$, there exists $v \in \mathbb{F}_\pi^{N\times 1}$, $v \neq 0$ such that $Mv = 0$. Then we have $c_0 v = -\sum_{i=1}^{\deg M} c_i M^i v = 0$, so $c_0 = 0$. Hence we can write $C(X) = X^c h(X)$ with $c \in \mathbb{N}^*$ and where $h(X)$ is not a vanishing polynomial for M. Note that $h^+(X) := Xh(X)$ is a vanishing polynomial of M. If v is any vector not included in the kernel of $h(M)$, and if we put $w = h(M)v$ then we have

$$w \neq 0 \text{ and } Mw = h^+(M)v = 0. \tag{6.17}$$

Hence, Wiedemann's algorithm reduces the computation of a solution v to that of the computation of any vanishing polynomial C of M. For this latter task, a necessary subroutine is the computation of a linear generator defined as follows.

Definition 6.3.1. Let K be a field and $(a_i)_{i\in\mathbb{N}}$ a sequence of elements in K. We say that a polynomial $\Lambda(X) \in K[X]$ is a linear generator of (a_i) if the formal series $(\sum_{i=0}^{\infty} a_i X^i)\Lambda(X)$ is a polynomial. If Λ has a non-zero constant coefficient and we call N' its degree, $\Lambda(X) = \sum_{i=0}^{N'} \Lambda_i X^i$, the condition is equivalent to

$$\forall n \geq N' + 1, \qquad a_n = \frac{-1}{\Lambda_0} \sum_{i=1}^{N'} a_{n-i}\Lambda_i. \tag{6.18}$$

Note that one can define *the* linear generator of a sequence as the unique monic linear generator Λ of minimal degree. Also note that, given a sequence which is known to have a degree N' linear generator, one can determine a linear generator from any $2N'$ consecutive terms. Indeed, let us suppose that we know $a_{n_0-N'}$, $a_{n_0-N'+1}$, ..., $a_{n_0+N'-1}$ for some n_0. Then, we write Equation (6.18) for $n = n_0, n_0+1, \ldots, n_0+N'-1$. We impose $\Lambda_0 = 1$ and we obtain a linear system of size $N' \times N'$ whose unknowns are the coefficients of a linear generator Λ. Unfortunately, this naive technique is impractical since it has a cost of $O(N'^3)$.

A faster method for computing the linear generator was proposed by Berlekamp and Massey (see Programme 8.1 in [Tho03]), and has a complexity of $O(N'^2)$ multiplications in $\mathbb{F}_\pi$. It consists essentially in applying the Extended Euclid Algorithm (EEA) to $a = \sum_{i=0}^{2N'-1} a_i X^i$ and $b = X^{2N'}$ and interrupting it in the middle. Indeed, EEA computes iteratively triplets (d_i, s_i, t_i) of polynomials such that

$$d_i = s_i a + t_i b. \tag{6.19}$$

The degree of d_i can be guaranteed to decrease by exactly 1 at each iteration. At the same time, the degree of s_i increases of at most 1 at each step. We interrupt EEA when $\deg d_i = N' - 1$ so that $\deg s_i \leq N'$. When we write Equation (6.19) for the value of i at the interruption time we obtain

$$s_i a \equiv d_i \mod X^{N'}. \tag{6.20}$$

This is equivalent to the linear system which determines the linear generator up to a scalar. Hence s_i is a linear generator of the sequence (a_i).

We can now summarize the linear algebra stage in Algorithm 6.4.

Algorithm 6.4 The linear algebra stage: Wiedemann

Input: An $N \times N$ singular matrix M over a field K

Output: a non-trivial solution of $Mu = 0$

1: x←Random($K^{N\times 1}$), y←Random($K^{1\times N}$),
2: [Krylov] Compute $a_i = yM^i$x for i in $[0, 2N-1]$
3: [Linear generator] Compute the linear generator $\Lambda = \sum_{i=0}^{\deg\Lambda} c_{\deg\Lambda - i} X^i$ of $(a_n)_{n\in\mathbb{N}}$
4: $C(X) \leftarrow \sum_{i=0}^{\deg\Lambda} c_i X^i$
5: $h(X) \leftarrow X^{-\operatorname{val}_X C} C(X)$
6: v ←Random($K^{N\times 1}$); ▷ can be $v \leftarrow$ x
7: [Make solution] $u \leftarrow h(M)v$
8: Repeat $u \leftarrow Mu$ until $Mu = 0$ and $u \neq 0$ ▷ at most $N + 1 - \deg\Lambda$ iterations

Let us explain the correctness of the algorithm. The polynomial C in Algorithm 6.4 is such that

$$y\left(\sum_{i=0}^{\deg C} c_i M^i\right) x = 0. \tag{6.21}$$

One can show that $\sum_{i=0}^{\deg C} c_i M^i = 0$ except if x, y or v are in the kernel of $Q(M)$ for a divisor Q of the characteristic polynomial of M. Hence, the probability of failure is comparable to $1/\#K$ (see a detailed study in [Tho03]). Coming back to the particular case $K = \mathbb{F}_\pi$ we conclude that, the linear generator Λ allows to obtain a vanishing polynomial of M with probability $1 - O(1/\pi)$. Contrary to the factorization context, the failing cases in Wiedemann's algorithm are not an issue in algorithms of discrete logarithm where π is very large.

The complexity of each matrix-vector product is $N\lambda$, where λ is the average number of non-zero entries per row. A dot product costs N multiplications and N additions in the field. Hence, the Krylov step in line 2. has a cost of $(2 + o(1))N^2\lambda$ operations. We saw that the linear generator can be computed in $O(N^2)$ operations, which is negligible with respect to $N^2\lambda$ when λ grows with N. Finally, the cost of making a solution is dominated by the reevaluation[3] of $M^i v$ for i in $[0, N]$. This takes another $(1 + o(1))N^2\lambda$ operations. We conclude that

$$\text{time (linear algebra stage)} = \big(3 + o(1)\big)\, N^2\lambda. \tag{6.22}$$

[3]Storing the totality of the sequence $M^i x$ is usually unfeasible.

6.3.5 Individual logarithm

Given an element s of $\mathbb{F}_p^*$ and the virtual logarithms of all the elements in the factor base with respect to a generator t of $\mathbb{F}_p^*$, the individual logarithm stage consists in searching for $\log_t s$. We assume in this section that the polynomial g is linear and we discuss in Section 8.4 the necessary modifications in the non-linear case. In the first step of the individual logarithm stage, called *smoothing*, we find an integer $h \in [1, p-2]$ such that $(t^h s \bmod p)$ is $L_p(2/3, c)$-smooth for a positive constant c. The best choice of the constant c was analyzed in Chapter 4. In the second step of the individual logarithm stage, called *descent* (also called reduction in [Sem02],[CS06]), one expresses any desired virtual logarithm as a sum of virtual logarithms of smaller primes and prime ideals. Note that the primes can be seen as prime ideals of the rational side. The descent ends when all the primes and prime ideals are in the factor base, making possible to reconstruct the discrete logarithm of every prime in the factorization of $(t^h s \bmod p)$.

Let q be a prime dividing $(t^h s \bmod p)$ for the value of h found during the *smoothing*. Tehe Q-lattice of q is the lattice

$$\mathrm{Lat}(q) = \left\{(a,b) \in \mathbb{Z}^2 \mid a - mb \equiv 0 \bmod q\right\}. \tag{6.23}$$

We consider a constant λ in the interval $(0,1)$. Using ECM or by sieving, depending on the size of q, as we explain in Section 7.3, one finds a pair (a,b) in $\mathrm{Lat}(q)$ such that

- $(a-bm)/q$ is q^λ-smooth;
- $\mathrm{N}(a-b\alpha)$ is q^λ-smooth.

This subroutine is called a descent step. Due to Equation (6.14) (later refined in Equation (6.36)) we can write $\log_t q$ as a linear combination of Schirokauer maps, discrete logarithms and virtual logarithms of smaller primes or prime ideals of smaller norm.

We proceed in the same manner in order to further descend all the primes introduced by q. For any prime ideal $\mathfrak{q} = \langle q_1, \alpha - \rho\rangle$ we consider the lattice

$$\mathrm{Lat}(\mathfrak{q}) = \left\{(a,b) \in \mathbb{Z}^2 \mid a - b\rho \equiv 0 \bmod q_1\right\}. \tag{6.24}$$

We use a coprime pair (a,b) such that $\mathrm{N}(a-b\alpha)/q_1$ and $(a-bm)$ are q_1^λ-smooth.

Hence one obtains a descent tree, where all primes or prime ideals descend to primes or prime ideals of smaller norm. When all the leaves of the descent tree are in the factor base, we obtain $\log_s t$ as a linear combination of the discrete logarithms, respectively virtual logarithms, of the elements in the factor base.

It is important to note that all the ideals which occur in the tree either have degree 1 or divide the conductor of f. Indeed, since we use coprime pairs (a,b), Proposition 6.1.4 guarantees that all prime ideals in $(a-b\alpha)$ have degree 1. This equally applies when both f and g are non-linear.

Let us evaluate the number of nodes in the descent tree. Each prime or prime ideal, which descends due to a pair (a,b), introduces at most $\log_2 F(a,b) + \log_2 |a - bm|$ new nodes. The analysis done in Section 7.3 shows that $\mathrm{N}(a-b\alpha)$ and $|a-bm|$

are less than p, so the arity of the tree is less than $2\log_2 p$. Its depth is the least integer w such that

$$\lambda^w \log_2 L_p(2/3, c) \leq \log_2 B. \tag{6.25}$$

Hence $w = O(\log\log p)$ and the number of nodes in the tree is $\exp\left((\log\log p)^2\right)$. Let us call T the complexity of each descent step. Under the reserve of proving that T is a sub-exponential function of $\log p$, we conclude that

$$\text{time (descent)} = T^{1+o(1)}. \tag{6.26}$$

The computation of T differs among the variants of NFS, so we leave it for Section 7.3.

6.4 Virtual logarithms

We showed how to transform a relation into an additive equation using a series of assumptions. In this section we show how this define virtual logarithms and how they allow to "take logarithms" in multiplicative relations. Let K be a number field and π a large prime. It is reasonable to consider that $\text{Disc}(f)$ is not divisible by π for some defining polynomial f of K. This guarantees that K has no πth roots of unity (see Lemma 6.1.7). We also suppose in this section–except in 6.4.4–that the class number h of K is coprime to π.

6.4.1 Defining virtual logarithms

Before showing how to use the virtual logarithms in the number field sieve, we must define them formally. This requires to introduce and compute the so-called Schirokauer maps.

Definition 6.4.1. Let π be a prime, let K be a number field with $\gcd(\text{Disc}(K), \pi) = 1$, $\mathcal{O}$ its ring of integers and call r the rank of its group of units $\mathcal{O}^*$. Put $K_\pi = \{\gamma \in K^* \mid \text{val}_\pi(\text{N}(\gamma)) = 0\}$. We call Schirokauer map any linear application

$$\Lambda : K_\pi/(K_\pi)^\pi \to (\mathbb{Z}/\pi\mathbb{Z})^r \tag{6.27}$$

which has full rank on $\mathcal{O}^*$, i.e., $\{\Lambda(u) \mid u \in \mathcal{O}^*\}$ equals $(\mathbb{Z}/\pi\mathbb{Z})^r$.

As an example, recall that, for any element γ of K^* one can write

$$\gamma^h = u' \prod_{i=1}^{r} u_i^{v(\gamma,u_i)} \prod_{\mathfrak{l} \text{ prime ideal}} \gamma_\mathfrak{l}^{v(\gamma,\mathfrak{l})}, \tag{6.28}$$

where u' is a root of unity, the elements u_i form a system of fundamental units and where, for each prime ideal $\mathfrak{l}$, $\gamma_\mathfrak{l}$ is a generator of $\mathfrak{l}^h$. One can easily check that, for fixed values of $\gamma_\mathfrak{l}$, u' and the exponents of the units u_i are uniquely defined. Moreover, the map Λ given by $\Lambda(\gamma) = \left(v(\gamma, u_i) \bmod \pi \mid i \in [1, r]\right)$ is a Schirokauer map. But recall that it is hard to compute a system of fundamental units.

Definition 6.4.2. Let p be a prime, π be a prime divisor of $p-1$ and let t be an element of $\mathbb{F}_p^*$ whose order is divisible by π. Let K be a number field containing a prime ideal $\mathfrak{p}$ of degree 1 over p. For any element γ in K_π we write $\overline{\gamma}$ for the reduction of γ modulo $\mathfrak{p}$. Let Λ be a Schirokauer map associated to K and π. For each prime ideal $\mathfrak{l}$ of $\mathcal{O}$, coprime to $p\mathcal{O}$, we fix an element $\gamma_\mathfrak{l} \in K^*$ such that

- $\gamma_\mathfrak{l}$ generates $\mathfrak{l}^h$;
- $\Lambda(\gamma_\mathfrak{l}) = 0$.

Then we define the virtual logarithm of $\mathfrak{l}$ with respect to t and Λ as

$$\log_{t,\Lambda}(\mathfrak{l}) = h^{-1} \log_t \overline{\gamma_\mathfrak{l}} \mod \pi. \tag{6.29}$$

6.4.2 Computing Schirokauer maps

Let K be a number field and π a prime. Call $\mathcal{O}$ the ring of integers in K. Let $P_1, \ldots, P_k$ be the prime ideals in the decomposition of π, which we assumed to be unramified (for an effective computations see Proposition 6.1.2):

$$\pi\mathcal{O} = \prod_{i=1}^{k} P_i. \tag{6.30}$$

For each i in $[1,k]$, put $m_i = [\mathcal{O}/P_i : \mathbb{F}_\pi]$, the inertia degree of P_i. Next, put $m = \operatorname{lcm}\big(m_i \mid i \in [1,k]\big)$ and $e = \pi^m - 1$. We define the Λ map as a function with values in $(\mathbb{Z}/\pi\mathbb{Z})^{[K:\mathbb{Q}]}$ but it is convenient to see it as a procedure which first maps an element of K in $\pi\mathcal{O}/\pi^2\mathcal{O}$ and then further brings it in $(\mathbb{Z}/\pi\mathbb{Z})^{[K:\mathbb{Q}]}$:

$$\Lambda : \left\{ \begin{array}{ccccc} \{\gamma \in K^* \mid \operatorname{val}_\pi \mathrm{N}(\gamma) = 0\} & \to & \pi\mathcal{O}/\pi^2\mathcal{O} & \to & (\mathbb{Z}/\pi\mathbb{Z})^{[K:\mathbb{Q}]} \\ \gamma & \mapsto & \gamma^e - 1 & \mapsto & \frac{1}{\pi}(\gamma^e - 1). \end{array} \right. \tag{6.31}$$

As before, let $f \in \mathbb{Z}[x]$ be a defining polynomial for K such that π is coprime to $\mathrm{Disc}(f)$ and to the leading coefficient c_d of f. Call α a root of f in K. Let γ be an element for which we want to compute $\Lambda(\gamma)$. Without loss of generality, one can assume that γ belongs to $c_d^{\deg f}\mathbb{Z}[\alpha] \subset \mathcal{O}$. Indeed, any element γ of K can be written as $\gamma = \frac{\gamma_1}{\gamma_2}$ with γ_1 in $c_d^{\deg f}\mathbb{Z}[\alpha]$ and $\gamma_2 \in \mathbb{Z} \subset \mathbb{Z}[\alpha]$, so that one can obtain $\Lambda(\gamma)$ as $\Lambda(\gamma_1) - \Lambda(\gamma_2)$. We compute γ^e using a fast exponentiation technique in $\mathbb{Z}[x]/\langle \pi^2, f\rangle$. Hence we obtain the coordinates of $\gamma^e - 1$ in $\mathbb{Z}[\alpha]/\pi^2\mathbb{Z}[\alpha]$. Since π is coprime to $\mathrm{Disc}(f)$ and c_d, they are the coordinates of $\gamma^e - 1$ in a basis of $\mathcal{O}/\pi^2\mathcal{O}$. When dividing them modulo π, we obtain the $[K:\mathbb{Q}]$ components of Λ.

We claim that in most cases the map Λ is a Schirokauer map. Indeed, we have no easy way to test the rank condition for a given map λ because no fast algorithm of computing a generating set of $\mathcal{O}^*$ is known. On the contrary, Λ is always linear, as shown by the following result.

Lemma 6.4.3. *Let Λ be defined as in Equation (6.31). Then Λ is well defined and, for all γ_1 and γ_2 in K_π, we have*

$$\Lambda(\gamma_1\gamma_2) = \Lambda(\gamma_1) + \Lambda(\gamma_2). \tag{6.32}$$

Proof. Let γ be an element of $\mathcal{O}$ whose norm is coprime to π. For all $i \in [1,k]$, the residue of γ modulo P_i is a non-zero element of the residue field of P_i. This latter field has cardinality π^{m_i}, so we have $\gamma^e \equiv 1 \bmod P_i$. As π is unramified, we have $\pi\mathcal{O} = \prod_{i=1}^{k} P_i$, so $\gamma^e \equiv 1 \mod \pi\mathcal{O}$. This shows that Λ is well defined.

Let γ_1 and γ_2 be two elements whose norms are coprime to π. Let δ_1 and δ_2 be two elements of $\mathcal{O}/\pi\mathcal{O}$ such that, for $i \in [1,2]$, $\gamma_i^e - 1 \equiv \pi\delta_i \bmod \pi^2$. The result follows from the following equation

$$(\pi\delta_1 + 1)(\pi\delta_2 + 1) \equiv \pi(\delta_1 + \delta_2) + 1 \mod \pi^2. \tag{6.33}$$

□

Note that Λ has $[K : \mathbb{Q}]$ coordinates. If r is the rank of $\mathcal{O}^*$, one can extract r coordinates $(\lambda_1, \ldots, \lambda_r)$ without reducing the rank of Λ.

We emphasize that, despite the lack of a fast algorithm to test the rank condition, we expect it to be virtually always true. Indeed, Schirokauer [Sch05] gave heuristic arguments that the probability that Λ has full rank is $1 - O(1/\pi)$, when π goes to infinity.

6.4.3 Using virtual logarithms in NFS

Let Λ be a Schirokauer map, for example the one computed above. Since the rank of $(\lambda_1, \ldots, \lambda_r)(\mathcal{O}^*)$ is full, there exist units u_i for $i \in [1,r]$ such that

$$\begin{aligned} \lambda_i(u_i) &= h^{-1} \bmod \pi \\ \forall j \neq i, \lambda_j(u_i) &= 0. \end{aligned}$$

Note in particular that (u_i) is a system of fundamental units in K, which cannot be computed, but which helps us give a theoretical meaning of the coordinates of Λ as discrete logarithms.

We claim that, if $\Lambda(u) = 0$ for a unit $u \in \mathcal{O}^*$, then u is a root of unity multiplied by a πth power. Indeed, write $u = u_0 \prod_{i=1}^{r} u_i^{e_i}$ for a root of unity u_0 and integers e_i. If ϵ is the order of u_0, then $\epsilon\Lambda(u_0) = 0$. Since, K has no πth roots, $\gcd(\epsilon, \pi) = 1$, so $\Lambda(u_0) = 0$. Then $\Lambda(u) = (e_1 h^{-1} \bmod \pi, \ldots, e_r h^{-1} \bmod \pi)$, so all the exponents e_i are multiples of π.

For each prime ideal $\mathfrak{l}$, let $\gamma_\mathfrak{l}$ be a generator of $\mathfrak{l}^h$ such that $\Lambda(\gamma_\mathfrak{l}) = 0$. Then, for any $\gamma \in \mathcal{O}$, with $\mathrm{val}_\pi \mathrm{N}(\gamma) = 0$ we have

$$\gamma^h = u \prod_{i=1}^{r} u_i^{h\lambda_i(\gamma)} \prod_{\mathfrak{l} \text{ prime ideal}} \gamma_\mathfrak{l}^{v(\gamma,\mathfrak{l})}, \tag{6.34}$$

where u is an element of $\mathcal{O}$, λ_i is the i^{th} coordinate of Λ and $v(\gamma, \mathfrak{l})$ is the valuation of γ at $\mathfrak{l}$. Since u has zero valuation at any prime ideal, it belongs to $\mathcal{O}^*$. A direct computation shows that $\Lambda(u) = 0$ and then we can write $u = u_0\omega^\pi$ for a root of unity u_0 and an integer element ω of K.

Since the order of u_0 is coprime to π, the same is true for the order of its image $\overline{u_0}$ in $\mathbb{F}_p$, hence $\log \overline{u_0} \equiv 0 \bmod \pi$ regardless of the basis of the logarithm. Further

$\log_t \overline{u} \equiv 0 \bmod \pi$. By sending Equation (6.34) in $\mathbb{F}_p^*$ and taking discrete logarithms modulo π, we obtain the additive equation

$$\log_t \overline{\gamma} \equiv \sum_{i=1}^{r} \lambda_i(\gamma) \log_t \overline{u_i} + \sum_{\mathfrak{l} \text{ prime ideal}} v(\gamma, \mathfrak{l}) \log_{t,\Lambda}(\mathfrak{l}) \mod \pi, \tag{6.35}$$

where $\overline{\gamma}$ and $\overline{u_i}$, $i \in [1, r]$, stand for the residues of γ and u_i in $\mathbb{F}_p$. If (a, b) is a doubly-smooth pair for NFS, of number fields $\mathbb{Q}(\alpha_f)$ and $\mathbb{Q}(\alpha_g)$ having unit rank r_f and r_g, then we can write an additive equation as follows:

$$\begin{aligned} &\textstyle\sum_{i=1}^{r_f} \lambda_i^{(f)}(a - b\alpha_f) \log_t \overline{u_i} + \sum_{\mathfrak{l} \text{ prime ideal}} v(a - b\alpha_f, \mathfrak{l}) \log_{t,\Lambda_g}(\mathfrak{l}) \equiv \\ &\qquad \equiv \textstyle\sum_{i=1}^{r_g} \lambda_i^{(g)}(a - b\alpha_g) \log_t \overline{u_i} + \sum_{\mathfrak{l} \text{ prime ideal}} v(a - b\alpha_g, \mathfrak{l}) \log_{t,\Lambda_g}(\mathfrak{l}) \bmod \pi \end{aligned} \tag{6.36}$$

Note that Equation (6.36) generalizes Equation (6.14).

6.4.4 Removing the class number condition

One usually makes the heuristic assumption that the class number of K is coprime to π. Note that it is a critical prerequisite for the definition of the virtual logarithms. In this section we show that even when this condition fails we expect the computations in NFS to succeed.

Let K be a number field, π a prime and Λ a Schirokauer map with respect to K and π. Let $\mathcal{F}$ be the (infinite) set of all the degree 1 prime ideals which lie above primes other than π. As before K_π is the group of elements in K^* of norm coprime to π. Let us define

$$\psi : \begin{cases} K_\pi/(K_\pi)^\pi & \to & (\mathbb{Z}/\pi\mathbb{Z})^r \times (\mathbb{Z}/\pi\mathbb{Z})^{\mathcal{F}} \\ \gamma & \mapsto & \left(\Lambda(\gamma), (\mathrm{val}_{\mathfrak{l}}\, \gamma, \mathfrak{l} \in \mathcal{F})\right), \end{cases} \tag{6.37}$$

where r is the rank of the group of units in K.

Let now assume that K is as in the NFS algorithm, i.e., there exists a degree 1 prime ideal above p. Let t be a generator of $\mathbb{F}_p^*$. We define

$$\mathfrak{L} : \begin{cases} \mathrm{Im}(\psi) \subset (\mathbb{Z}/\pi\mathbb{Z})^r \times (\mathbb{Z}/\pi\mathbb{Z})^{\mathcal{F}} & \to & \mathbb{Z}/\pi\mathbb{Z} \\ w & \mapsto & \log_t \overline{\gamma}, \end{cases} \tag{6.38}$$

where γ is the unique element of $K_\pi/(K_\pi)^\pi$ such that $\psi(\gamma) = w$ and where $\overline{\gamma}$ is its image in $\mathbb{F}_p$. To see that γ is unique remember that if a unit u is such that $\Lambda(u) = 0$ then u is a root of unity multiplied by a πth power and use the assumption that π is coprime to $\mathrm{Disc}(f)$ so that K has no πth roots of unity.

Let us call $\mathfrak{L}_1$, $\mathfrak{L}_2$, ψ_1 and ψ_2 the $\mathfrak{L}$ and ψ maps corresponding to the two number fields in NFS. Each pair (a, b) found in the relation collection stage of NFS yields an equation as follows

$$\mathfrak{L}_1(w_1) = \mathfrak{L}_2(w_2), \tag{6.39}$$

where $w_i = \psi_i(a - b\alpha_i)$ for $i \in [1, 2]$. We make the heuristic assumption that the equations hence formed yield a matrix of full rank. In this case, since $\mathfrak{L}_1$ and $\mathfrak{L}_1$ are linear, in the linear algebra stage of NFS one computes $\mathfrak{L}_1$ and $\mathfrak{L}_2$. The discrete logarithm of any smooth element can then be computed using $\mathfrak{L}_1$ and $\mathfrak{L}_2$.

6.5 Computing valuations at problematic primes

We saw that, when f is monic and when the factor base contains no prime or prime ideal dividing the conductor of $\mathbb{Z}[\alpha]$ (resp. $\mathbb{F}_q[t][\alpha]$), one can construct the matrix using Dedekind's method (Proposition 6.1.4). The condition that f is monic is too restrictive in practice where one wants to select the polynomials which optimize the efficiency of the sieving stage. Moreover, if the conductor has small divisors, e.g. 2 (resp. t), then one will loose a large fraction of the doubly-smooth pairs if one eliminates the problematic primes from the factor base.

In order to compute valuations in the general case, one uses classical algorithms for which we refer to [Coh93]. We make here a short presentation of the manner in which these algorithms must be combined in NFS and FFS.

The natural question is what are the primes (resp. irreducible polynomials) for which one can use Dedekind's method. Call f_d the leading coefficient of f and note that $\theta := f_d\alpha$ is an integer of K. We split the primes (resp. irreducible polynomials) in four categories:

1. coprime to $f_d \operatorname{Disc}(f)$.
2. ramified, i.e., divisors of $\operatorname{Disc}(K)$ and, in this section, coprime to f_d;
3. bad, i.e., divisors of the conductor of f and, in this section, coprime to f_d;
4. divisors of f_d;

In this section ℓ is a prime or an irreducible polynomial; for convenience, all the results of this section are stated in the NFS context.

Generic ℓ. Let ℓ be a prime (resp. irreducible polynomial) of the first type. We put $\theta = f_d\alpha$ and note that θ is an integer of K. Let (a, b) be a coprime pair. Set a' and b' such that $af_d = a'g$ and $b = b'g$ with $g = \gcd(f_d, b)$. Note that one can apply Dedekind's method (see Proposition 6.1.4) to $a' - b'\theta$ for all prime ideals above ℓ. Since $a' - b'\theta$ is $a - b\alpha$ multiplied by a divisor of f_d, they have the same prime ideal divisors and have equal valuations above all primes (resp. irreducible polynomials) which do not divide f_d. Hence, in NFS it is enough to apply Dedekind's method to the pair $\left(\frac{af_d}{\gcd(f_d,b)}, \frac{b}{\gcd(b,f_d)}\right)$.

Ramified ℓ. Dedekind's method applies to ramified primes (resp. irreducible polynomials) which do not divide f_d (see Proposition 6.1.4). The difficulty is to distinguish ramified from bad primes (resp. irreducible polynomials).

We use as a reference Sections 4.8 and 6.2 in [Coh93]. By trial division one can determine the divisors of $\operatorname{Disc}(f)$ up to the smoothness bound B. Hence we have a combined list of bad and ramified primes (resp. irreducible polynomials) in the factor base. Let us see how to determine the type of a candidate ℓ. We call ℓ-maximal any order $\mathcal{O}_\ell$ such that $[\mathcal{O} : \mathcal{O}_\ell]$ is coprime to ℓ. This is computed by the Round 2 algorithm in polynomial time with respect to $\log \ell$ (resp. $\deg \ell$) and $\deg f$. Then ℓ is bad if and only if $\mathbb{Z}[\theta]$ (resp. $\mathbb{F}_q[t][\theta]$) is not ℓ-maximal.

Bad primes and divisors of f_d. Let us now study the general case. Let ℓ be a prime (resp. irreducible polynomial) and let us consider that we computed an ℓ-maximal order $\mathcal{O}_\ell$. One can impose that $\mathcal{O}_\ell$ is a sub-ring of $\mathcal{O}$ as the construction of $\mathcal{O}_\ell$ is done by enlarging the equation order in $\mathcal{O}$. This is represented by a $\mathbb{Z}$-module (resp $\mathbb{F}_q[t]$-module) basis of $\mathcal{O}_\ell$, each element being given by its coordinates in basis $(1, \alpha, \alpha^2, \ldots, \alpha^{\deg f-1})$. A key remark is that there is a bijection between the prime ideals in the decomposition of $\ell\mathcal{O}_\ell$ and the prime ideals in the decomposition of $\ell\mathcal{O}$. Moreover, for any ideal I in $\mathcal{O}_\ell$ and a prime ideal $\mathfrak{l}$ of $\mathcal{O}_\ell$, one has:

$$\mathrm{val}_{\mathfrak{l}}\, I = \mathrm{val}_{(\mathfrak{l}\mathcal{O})}(I\mathcal{O}). \tag{6.40}$$

One decomposes into prime ideals the ideal $\ell\mathcal{O}_\ell$ in polynomial time using Buchmann-Lenstra's method (Algorithm 6.2.9 in [Coh93]) with $\mathcal{O}_\ell$ instead of $\mathcal{O}$. The result the list of prime ideals dividing $\ell\mathcal{O}_\ell$ and theirs valuations in ℓ. Each ideal is given by a $\mathbb{Z}$-module (resp. $\mathbb{F}_q[t]$-module) basis formed of elements of $\mathcal{O}_\ell \subset K$. Also in polynomial time, by Algorithm 4.8.17 in [Coh93], for each prime ideal $\mathfrak{l}$ above ℓ, we compute an element $\pi_{\mathfrak{l}}$ such that

$$\pi_{\mathfrak{l}} \in K \backslash \mathcal{O}_\ell \text{ and } \pi_{\mathfrak{l}}\mathfrak{l} \subset \mathcal{O}_\ell. \tag{6.41}$$

By Lemma 4.8.16 in [Coh93], for all ideal I of $\mathcal{O}$ we have $\mathrm{val}_{\mathfrak{l}\mathcal{O}_\ell}(I\mathcal{O}_\ell) = \max\{k \in \mathbb{N} \mid \pi_{\mathfrak{l}}^k I \subset \mathcal{O}_\ell\}$. Hence, the element $\pi_{\mathfrak{l}}$ associated to $\mathfrak{l}$ allows to compute the valuation of $\mathfrak{l}$ at any ideal of $\mathcal{O}$, and therefore at $(a - b\alpha)$ for any pair (a, b) occurring in the computations.

Note that the passage from $(a - \alpha b)$ to an ideal of $\mathcal{O}$ can be trivially obtained when multiplying by f_d. Nevertheless, one can prefer to multiply $(a - b\alpha)$ by the ideal $J = \langle f_d, f_d\alpha + f_{d-1}, \ldots, \sum_{i=1}^{d} f_i\alpha^{i-1}\rangle$. Indeed, one can show that J is an ideal of $\mathcal{O}$ of norm f_d and that, for all pairs (a, b), $J_{a,b} := (a - b\alpha)J$ is an ideal of $\mathcal{O}$ [Mon97]. Since the norm of $J_{a,b}$ is smaller than the norm of $f_d(a - b\alpha)\mathcal{O}$, one obtains a matrix with slightly less entries per row.

Algorithm. To sum up, one needs to prepare the matrix construction by making a list of bad primes (resp. irreducible polynomials) and divisors of f_d. Also in the pre-computations, one decomposes into prime ideals all these primes (resp. irreducible polynomials) and, for each prime ideal $\mathfrak{l}$ above bad primes or prime divisors of f_d (resp. irreducible polynomials), one finds an element $\pi_\ell \in K$ as in Equation (6.41). The actual matrix construction either uses Dedekind's method or uses the pre-computed elements $\pi_{\mathfrak{l}}$. Hence the time-dominating step in the matrix construction is the factorization of $F(a, b)$ and $G(a, b)$. We give a precise presentation in Algorithm 6.5. Note that, for simplicity, we assume that g is linear and m is its root modulo p.

6.6 The Function Field Sieve

6.6.1 Differences with NFS

The function field sieve is very similar to the number field sieve if analyzed at a high level, but at a closer look some stages are easier. Indeed, on the one

Algorithm 6.5 Constructing the matrix

Input: the smoothness bound B and a list of doubly-B-smooth coprime pairs (a, b)

Output: the valuations vector of $(a - b\alpha)$ and $(a - bm)$

Make a list L of bad primes of f and divisors of f_d up to B
for ℓ in L **do**
    Compute an ℓ-maximal order $\mathcal{O}_\ell$;
    Find a basis and the ramification degree of each prime ideal above ℓ;
    for $\mathfrak{l}$ prime ideal above ℓ **do**
        Find an element $\pi_\mathfrak{l}$ as in Equation (6.41)
    end for
end for
for pair (a, b) **do**
    Factor $F(a, b)$ and $a - bm$
    Compute the valuation vector of $a - bm$
    for ℓ prime divisor of $F(a, b)$ **do**
        if $\ell \notin L$ **then**
            Compute valuations of $a - b\alpha$ with Dedekind's method
        else
            Compute valuations of $a - b\alpha$ using the elements $\pi_\mathfrak{l}$
        end if
    end for
end for

hand many computations on integer numbers can be translated to the context of polynomials, e.g. multiplication, Euclidean algorithm, sieving etc. On the other hand, some algorithms are faster in this new context, the most noticeable being the factorization. Hence one can obtain an algorithm of same complexity as NFS by translating each step in terms of polynomials (see [Adl94]). But the most important advantage for FFS is that any given field $\mathbb{F}_{q^k}$ can be represented as $\mathbb{F}_q[t]/\langle\varphi\rangle$ for any irreducible polynomials $\varphi \in \mathbb{F}_q[t]$ of degree k (see [AH99],[JL02]). We will see that this reduces the complexity of the algorithm (see Section 7.5). After making a list of the differences of FFS with NFS, we will make an explicit presentation of each step of the FFS algorithm.

Polynomial selection Recall the base-m method of polynomial selection in Algorithm 6.2. This method can be translated to the case of FFS. Indeed, let $\mathbb{F}_{q^k}$ be a field whose elements are represented as residue classes in $\mathbb{F}_q[t]/\langle\varphi\rangle$ for an irreducible polynomial φ of degree k. One mimics the base-m method for a parameter d: one picks a polynomial $m(t) \in \mathbb{F}_q[t]$ of degree d; one computes the base-m representation of $\varphi(t)$: $\sum_{i=0}^{\lfloor k/d \rfloor} f_i(t)m(t)^i$ and one outputs $f = \sum_{i=0}^{\lfloor k/d \rfloor} f_i(t)x^i$ and $g = x - m(t)$.

An alternative proposed by Joux and Lercier [JL02] consists in selecting the two polynomials of FFS, f and g, before choosing the representation $\mathbb{F}_q[t]/\langle\varphi\rangle$ of the finite field $\mathbb{F}_{q^k}$. Indeed, let $\mathbb{F}_{q^k}$ be a given finite field $\mathbb{F}_{q^k}$ as the set of residue classes modulo an irreducible polynomial φ_0 and let T_0 and S_0 be two elements of $\mathbb{F}_{q^k}$ represented by two polynomials modulo φ_0. By computing a root t_0 of φ in $\mathbb{F}_q[t]/\langle\varphi\rangle$, one can determine a field isomorphism Ψ from $\mathbb{F}_q[t]/\langle\varphi_0\rangle$ to $\mathbb{F}_q[t]/\langle\varphi\rangle$. Hence we have the equality

$$\log_{T_0} S_0 = \log_{\Psi(T_0)} \Psi(S_0). \tag{6.42}$$

It is then sufficient to compute discrete logarithms is the field representation of our choice.

We give a detailed presentation of Joux and Lercier's method in Algorithm 6.6. We emphasize that the polynomial f has small coefficients in t, which can be put in analogy with the special number field sieve (cf. joke in the title of [JL02]).

Factorization and smoothness tests Due to polynomial-time probabilistic algorithms for factoring univariate polynomials, a series of problems which are computationally difficult for number fields can be solved in expected polynomial time in the case of the function fields. Recall in particular that, in the number field context each step of the computations of $\mathcal{O}_K$ is polynomial in time, except for the factorization of $\mathrm{Disc}(f)$. Hence the computation of $\mathcal{O}_K$, called assumption 1. in Section 6.2, can be done in the FFS context in polynomial time.

However, the most important speed-up corresponds to the complexity of the smoothness tests in the case of univariate polynomials. Indeed, the costly smoothness tests with ECM in the case of NFS can here be replaced by a simple factorization procedure. Moreover, one can use a faster smoothness test which does not

Algorithm 6.6 Polynomial selection in FFS: the Joux Lercier method

Input: a prime power q and an integer k; a parameter d

Output: a representation of $\mathbb{F}_{q^k}$ as $\mathbb{F}_q[t]/\langle\varphi\rangle$ and two monic irreducible polynomials f and g in $\mathbb{F}_q[t][x]$ such that

$$f(t_\varphi, x) \text{ and } g(t_\varphi, x) \in (\mathbb{F}_q[t]/\langle\varphi\rangle)[x] \text{ have a common root } m(t_\varphi),$$

where t_φ is a root of φ in $\mathbb{F}_q[t]/\langle\varphi\rangle$ and m is a polynomial in $\mathbb{F}_q[t]$.

1: Pick a polynomial f in $\mathbb{F}_q[t][x]$ with $\deg_x f = d$ and $\deg_t f$ small

2: Try random polynomials $g = g_0(t) - g_1(t)x$ with $\deg_t g = \lceil k/d \rceil$ until $\mathrm{Res}(f, g) = F(g_0, g_1)$ has an irreducible factor φ of degree k

3: Call t_φ a root of φ in $\mathbb{F}_{q^k}$ and choose m as a common root in $\mathbb{F}_{q^k}$ of the polynomials $f(t_\varphi, x)$ and $g(t_\varphi, x)$

4: Output f, g, φ and m

factor the candidate and which relies on the following identity

$$\forall \beta \in \mathbb{N}, t^{q^\beta} - t = \prod_{h \in H} h, \tag{6.43}$$

where H is the set of monic irreducible polynomials of degree dividing β. Then, for any bound β, one can test if a polynomial $P(t)$ is β-smooth as follows:

$$\text{compute } P'(t) \prod_{i=\lceil \beta/2 \rceil}^{\beta} \left(t^{q^i} - t\right) \mod P(t) \tag{6.44}$$

and declare that P is smooth if one obtains 0.

Eliminating the Schirokauer maps A further simplification which occurs when replacing the number fields with function fields is the existence of places at infinity. They will allow us to show that in the context of function fields, the assumption 3. in Section 6.2 can be replaced by a computationally easy task: determining the places at infinity. Due to its level of technicality, we leave this presentation for Section 6.6.3.

6.6.2 The FFS algorithm: stage by stage

Let us make an explicit presentation of the function field sieve, giving details on each point of analogy with the number field sieve.

Polynomial Selection One can select the polynomials f and g in $\mathbb{F}_q[t][x]$ using Algorithm 6.6. It is this algorithm that we study in detail and improve in Chapter 9. In particular we will argue that the time cost of this stage is not critical and that the linear polynomial g has little influence on the sieve. Hence, the important aspect of the polynomial selection is the choice of a polynomial f, for which we propose a specific analysis.

Relation collection Given two polynomials f and g in $\mathbb{F}_q[t][x]$, we collect coprime pairs (a, b) in $\mathbb{F}_q[t]^2$ such that $F(a, b)$ and $G(a, b)$ are β-smooth, where $F(X, Y) = f(X/Y)Y^{\deg f}$ and $G = g(X/Y)Y^{\deg g}$ are the homogenizations of f and g and where β is a parameter. As in the NFS algorithm we imposed that a is positive, here we impose that a is monic in order to avoid pairs (a, b) which are multiple of each other. For the algorithm note that we can take advantage of the smoothness test presented in Equation (6.44). The reader can find many more methods to accelerate the sieve in [DGV13].

Filtering and the linear algebra stage Filtering has virtually no modification with respect to the NFS algorithm for discrete logarithm [Bou13].

The linear algebra stage can be implemented more effectively in the FFS algorithm than in NFS. Indeed, one can divide a NFS matrix in, on the one hand the columns corresponding to valuations at prime ideals and, on the other hand, the columns corresponding to Schirokauer maps. The valuation columns contain small coefficients, smaller than a computer word, and most of them are 1 and -1. Indeed, they are computed as valuation of irreducible polynomials in polynomials of the form $F(a, b)$ where F is the homogenization of f and (a, b) are coprime integers. We make the heuristic that the integers of the form $F(a, b)$ behave as random integers. One can show that, the probability that a random integer has square factors larger than a parameter B_0 goes to 0 when B_0 goes to infinity. On the other hand, the Schirokauer columns contain integers modulo π, the modulus of the discrete logarithm. This usually is a prime of several computer words. In the case of FFS, the Schirokauer maps are replaced by valuations at the places at infinity, which behave as valuations at any other place or, by analogy, as valuations at prime ideals in the NFS case. The reader can find more details on this stage of the algorithm in [Tho03] and [Jel12].

Individual logarithm stage Given the virtual logarithms of all the elements in the factor base with respect to a base $t \in \mathbb{F}_{q^k}^*$ and given an element $s \in \mathbb{F}_{q^k}^*$, we want to compute $\log_t s$.

Recall that in the case of NFS, the first step of the individual logarithm stage is the smoothing. This step translates perfectly to FFS: randomly choose $h \in [1, \pi - 2]$ until $t^h s$ is C-smooth for $C = ck^{2/3}(\log k)^{1/3}$, with $c > 0$. One can check that

the resulting complexity is of type $L(1/3, \cdot)$ due to the smoothness probabilities. However, as the smoothness tests can be done in probabilistic polynomial time, the second constant of the complexity is smaller, inciting Thomé in [Tho03] to call this step the "ignition of the descent" (*amorce de la descente*).

The descent process is very similar to the corresponding step of the NFS algorithm. We illustrate it with the case when g is monic and linear. Given an irreducible polynomial q_1 we consider the pairs of coprime polynomials in $\mathrm{Lat}(q_1)=\{(a,b) \in \mathbb{F}_q[t]^2 \mid a - bm \equiv 0 \bmod q_1\}$ where m is the common root of f and g modulo φ. By direct trial or using the lattice sieve technique, depending on the degree of q_1, we find a pair (a,b) such that $(a-mb)/q_1$ and $F(a,b)$ are $\lambda \deg q_1$-smooth, where $\lambda > 0$ is a constant. For example one can set λ to its value in the corresponding step of the NFS algorithm. By this we write the discrete logarithm of q_1 as a linear combination of discrete logarithms of irreducible polynomials and of virtual logarithms of degree 1 ideals whose norm is a polynomial of degree less than $\lambda \deg q_1$. We then descend each degree 1 ideal $\mathfrak{q} = \langle q_1, x - \rho\rangle$ in the function field of f in a similar manner, by considering the pairs of polynomials in $\mathrm{Lat}(\mathfrak{q})=\{(a,b) \in \mathbb{F}_q[t]^2 \mid a - b\rho \equiv 0 \bmod q_1\}$. Given a pair (a,b) on $\mathrm{Lat}(\mathfrak{q})$ one writes the virtual logarithm of $\mathfrak{q}$ as linear combination of discrete logarithms of irreducible polynomials of degree at most $\lambda \deg \mathrm{N}(\mathfrak{q})$ and virtual logarithms of degree 1 ideals whose norm has degree at most $\lambda \deg \mathrm{N}(\mathfrak{q})$. We end the descent when we obtain $\log_t s$ as a linear combination of discrete and virtual logarithms of elements in the factor base.

6.6.3 Replacing Schirokauer maps by valuations at infinity

We will make use of a series of results from the theory of function fields, for which we use [Sti08] and [Lor96] as references. First, the set of places of a function field K corresponds to the set of prime ideals of the integer ring $\mathcal{O}$ together with a finite set of places, called places at infinity. Next, the order of the divisor class group equals the class number h of K.

Let us fix a divisor I of degree 1. Then, for each place P, there exists an element γ_P in K such that

$$\mathrm{div}(\gamma_P) = h\left(P - (\deg P)I\right). \tag{6.45}$$

We can now state the main result of this section.

Theorem 6.6.1. *Let $f \in \mathbb{F}_q[t][x]$ be an absolutely irreducible polynomial and let K be its associated function field. Call h the class number of K. For all places P of K we denote by γ_P an element of K such that $\mathrm{div}(\gamma_P) = h(P - (\deg P)I)$, where I is a degree 1 divisor. Then, for every non-zero element z of K one can write*

$$z^h = \kappa \prod_{P \text{ place of } K} \gamma_P^{v(z,P)}, \tag{6.46}$$

where $\kappa \in \mathbb{F}_q^*$ and $v(z,P)$ are integers.

Proof. For each place P of K we denote by $v(z,P)$ the valuation of P at z. By the choice of the elements γ_P, we obtain

$$\operatorname{div}\left(z^h/\left(\prod_{P \text{ place of } K} \gamma_P^{v(z,P)}\right)\right) = n_I I, \tag{6.47}$$

where n_I is an integer. Since the divisor in the equation is a principal one, its degree is 0; hence $n_I = 0$. Finally, according to Corollary I.1.19 in [Sti08], if an algebraic function has a zero divisor, then it is a constant $\kappa \in \overline{\mathbb{F}_q}^* \bigcap K$. Since f is absolutely irreducible, κ belongs to $\mathbb{F}_q^*$. □

Let us see how to use this theorem in the FFS algorithm. Call f and g the two polynomials in the algorithm and, for convenience, assume that $g = x - m(t)$ for some polynomial m. We call K the function field of f and α a root of f in K. In order to compute $v(a - b\alpha, P)$ in Theorem 6.6.1 for a prime ideals P of $\mathcal{O}$, one can use Dedekind's method (see Proposition 6.1.4). The valuations of places at infinity can be computed as valuations of ideals above t in the function field of $f(1/t, x)$.

Note that Matsumoto [Mat99] proposed the $\mathcal{C}_{a,b}$ family of curves, which avoids the computation of valuations at infinity. Indeed, these curves are known to have a unique place at infinity. Since any principal divisor has degree 0, the valuation at infinity can be computed from the valuations at the prime ideals of $\mathcal{O}$.

Let us use Theorem 6.6.1 to "uniquely factor functions into places". Let (a,b) be a coprime pair of polynomials in $\mathbb{F}_q[t]$ such that $F(a,b)$ and $a - bm$ are smooth. Then, by Theorem 6.6.1, we have

$$(a - b\alpha)^h = \kappa \prod_{P \text{ place of } K} \gamma_P^{v_1(a,b,P)} \tag{6.48}$$

$$(a - bm) = \kappa_2 \prod_{P \text{ monic irreducible in } \mathbb{F}_q[t]} P(t)^{v_2(a,b,P)}, \tag{6.49}$$

with κ_1 and κ_2 in $\mathbb{F}_q^*$.

Recall that $\mathfrak{p} := \langle \varphi, \alpha - m(t) \rangle$ is a prime ideal in K. We denote by a bar the reduction of polynomials in $\mathbb{F}_q[t]$ modulo φ and of elements of K modulo $\mathfrak{p}$. Note that the elements κ_1 and κ_2 are in $\mathbb{F}_q^* \subset \mathbb{F}_{q^k}^*$, so they are unchanged by the reduction modulo φ and $\mathfrak{p}$.

Since the reductions of $(a - b\alpha)$ and $(a - bm)$ in $\mathbb{F}_{q^k}$ are equal, when sending Equations (6.48) and (6.49) in $\mathbb{F}_{q^k}$, we obtain

$$\left(\kappa_1 \prod_{P \text{ place of } K} \overline{\gamma_P}^{v_1(a,b,P)}\right) = \left(\kappa_2 \prod_{P \text{ monic irred. in } \mathbb{F}_q[t]} \overline{\gamma_P}^{v_2(a,b,P)}\right)^h. \tag{6.50}$$

Since π is coprime to $q-1$, we have $\log_t \kappa_1 \equiv \log_t \kappa_2 \equiv 0 \bmod \pi$ for any generator t of $\mathbb{F}_{q^k}$. Then, when taking logarithms in Equation (6.50) we obtain Equa-

tion (6.51).

$$\sum_{P \text{ place of } K_1} v_1(a,b,P)\frac{\log(\overline{\gamma_P})}{h} \equiv \sum_{P \text{ monic irred in } \mathbb{F}_q[t]} v_2(a,b,P)\log(\overline{P}) \mod \pi. \tag{6.51}$$

We conclude that the Schirokauer maps in Equation (6.36) can be replaced by valuations at infinity.

Chapter 7

Old and new complexities for NFS

We are now ready to state our main complexity results on NFS for discrete logarithm. On the one hand we point out that the results of Chapter 4 improve the complexity of the individual logarithm of NFS. On the other hand we propose a new variant of NFS called "the DL factory", similar to Coppersmith's factorization factory. The difficulty resides in the fact that we reduce the factor base so that the descent process is longer. For completeness, we include some results on the complexity of FFS, both in the general and the middle prime case.
The chapter is organized as follows. After the complexity analysis of the classical variant of NFS, we introduce the DL factory. We then make a list of the numerous variants of NFS: classical, SNFS, multi-field and DL factory. We conclude by presenting the complexity analysis of FFS.

7.1 Classical NFS over prime fields

Let us consider the basic variant of NFS, corresponding to the base-m polynomial selection, the line sieve and the Wiedemann algorithm. The parameters which influence the time of NFS are as follows:

- B = the smoothness bound;
- E = the sieve parameter, i.e., we sieve for pairs (a, b) in $[0, E] \times [-E, E]$;
- $d = \deg f$.

According to Chapter 6 the complexity of the sieve is $E^{2+o(1)}$ whereas the complexity of the linear algebra stage is $B^{2+o(1)}$. We optimize the parameters for the relation collection and linear algebra stage. We will see in Section 7.3 that, for the parameter values that we obtain, the individual logarithm stage is negligible when compared to the sieve and the linear algebra stage. Hence, the function to be minimized is:

$$\begin{aligned} \text{time(NFS)} &= B^{2+o(1)} + E^{2+o(1)} \\ &= \max(B, E)^{2+o(1)}. \end{aligned}$$

In order to minimize $\max(E, B)$ we note that E increases when B decreases and vice-versa. To see this requires a technical analysis. Indeed, on the one hand when

E decreases, the norms decrease in turn so we can further decrease E. Hence we can consider that the size of the norms is constant so that $E^2 = B/P_{\text{smooth}}(c, B)$ for some constant $c > 0$. Given the rapid decrease of $P_{\text{smooth}}(c, B)$ when B increases, the parameter E also decreases when B increases. Hence, we impose the following equality :

$$E = B. \tag{*}$$

Next, let us make the heuristic assumption that the norm of a pair in the sieving domain is smooth with the same probability as a random number of the same size. We equally assume that the probabilities to be smooth on the rational and the algebraic sides are independent. Hence, the probability that a pair is doubly-B-smooth equals the probability that two random numbers having the size of $\mathrm{N}(a - b\alpha)$ and $|a - bm|$ are simultaneously B-smooth. We denote by P the probability that a pair of the sieve domain is doubly-B-smooth. Then, in order to have enough equations for the linear algebra, we impose the inequality $2E^2P \geq 2B$. The fact that we end the sieving stage when the number of collected equations is just larger than the cardinality of the factor base translates in the following equation:

$$2E^2P = 2B, \tag{**}$$

where $2B$ is an upper bound for the cardinality of the factor base.

Let us estimate the probability P. The norm on the algebraic side is bounded as follows

$$\mathrm{N}(a - b\alpha) \leq \deg f \|f\|_\infty E^d.$$

Since $\|f\|_\infty \leq p^{1/d}$ and $E = B$, the algebraic norm is bounded by $p^{1/d}B^{d+o(1)}$.

The rational norm is $|a - bm|$ where m is less than $p^{1/d}$. Since $|a|, |b| \leq E = B$ we have

$$|a - bm| \leq 2Bp^{1/d} = B^{1+o(1)}p^{1/d}. \tag{7.1}$$

We can now write

$$\begin{aligned} P &= \rho\left(\frac{\log(p^{1/d}B^{1+o(1)})}{\log B}\right)\rho\left(\frac{\log(p^{1/d}B^{d+o(1)})}{\log B}\right) \\ &\geq \rho\left((1+o(1))\frac{\log(p^{2/d}B^{(d+1)(1+o(1))})}{\log B}\right)^{1+o(1)} \end{aligned}$$

Note that we used the fact that, for $x_1, x_2, y > 0$, we have $P_{\text{smooth}}(x_1, y)P_{\text{smooth}}(x_2, y) \geq P_{\text{smooth}}(x_1x_2, y)^{1+o(1)}$ which is trivially true (see also Corollary 1.1.2).

Now, for fixed B, we optimize the probability P by balancing the sizes of B^d and $p^{2/d}$. A rough estimation shows that, if $B = L_p(\theta, \cdot)$ for a constant θ in the interval $(0, 1)$, then one must take d such that $B^d = L(\frac{1+\theta}{2}, \cdot)$ and $P^{1/d} = L(\frac{1+\theta}{2}, \cdot)$. Then $P = L_p(\frac{1-\theta}{2}; \cdot)$ and Equation (**) implies that $\theta = 1/3$. Therefore we introduce the constants β and δ as follows

$$B = E^{1+o(1)} = L_p\left(1/3, \beta\right)^{1+o(1)} \quad \text{and} \quad d = \delta\left(\frac{\log p}{\log\log p}\right)^{1/3}.$$

We obtain $P = L_p\left(1/3, \frac{1}{3}(\delta + \frac{2}{\delta\beta})\right)^{-1}$. When injecting in Equation (**) we obtain

$$\beta = \frac{1}{3}(\delta + \frac{2}{\delta\beta}).$$

Since δ must be a real number we impose that the discriminant of the equation above is positive:

$$0 \leq \Delta = 9\beta^2 - \frac{8}{\beta}.$$

We minimize β, and hence the time of NFS, by taking $\beta = \sqrt[3]{8/9}$. This implies $\delta = \frac{3}{2}\beta = 3^{1/3}$. The complexity of NFS is $2B^{2+o(1)}$, so

$$\text{time(NFS)} = L_p\left(1/3, \sqrt[3]{64/9}\right)^{1+o(1)}. \tag{7.2}$$

7.2 Discrete logarithm factory

The DL factory is an adaptation of the idea of the Factorization factory to the computation of discrete logarithms in prime fields. This idea is to re-use the list of pairs (a, b) which are smooth on the rational side. In more detail, for all the primes of a given bit-size, i.e., $\lfloor \log p \rfloor$ is constant, we use the same integer $m_0 \approx p^{1/d}$. Next we prepare a permanent file with the coprime pairs (a, b) in a sieve domain of parameter E such that $a - bm_0$ is smooth. Given a prime p, we select a polynomial f such that $f(m_0) \equiv 0 \bmod p$. The sieve consists in using ECM to test the smoothness of the algebraic side of each pair (a, b) in the permanent file. The linear algebra and the individual logarithm stage are left unchanged.

Let us compute the optimal parameters. As in the classical version, E and d denote the sieve parameter and the degree of f. The smoothness parameter B corresponds to the rational side and we introduce a smoothness parameter C for the algebraic side. Since the rational and the algebraic side have different roles in the DL factory, B and C have *a priori* different values. We set positive constants β, γ, δ and ϵ such that $B = L_p(1/3, \beta)$, $C = L_p(1/3, \gamma)$, $E = L_p(1/3, \epsilon)$ and $d = \delta\left(\frac{\log p}{\log\log p}\right)^{1/3}$.

The optimality of the sieving domain translates into the equation

$$E^2 P_f P_g = (B + C), \tag{*}$$

where P_f and P_g are the smoothness probability of the algebraic and rational sides.

We also impose that the sieve and the linear algebra have equal costs. The linear algebra has a cost $(B + C)^2$ as in the classical variant. The complexity of the sieve has a new expression as we have to test smoothness on the algebraic side only. Since the smoothness test with a bound of size $L_p(1/3, c)$, with $c > 0$, costs $L_p(1/6, \cdot)^{1+o(1)}$, the time of the sieve equals the size of the permanent file up to a $1 + o(1)$ exponent. This latter file must contain P_f^{-1} more pairs than the cardinality of the factor base. This translates in

$$P_f^{-1}(B + C) = (B + C)^2. \tag{**}$$

Note that we must have $\beta = \gamma$. Indeed, if we had $\beta < \gamma$, then, by increasing β, we could reduce the time of the sieve without changing the time of the linear algebra except for an exponent $1 + o(1)$. If on the contrary, $\beta > \gamma$, then one can increase C and reduce E such that both Equations (**) and (*) are still satisfied. Hence we are left with the three parameters β, δ and ϵ.

One computes

$$P_f = L_p\left(1/3, \frac{1}{3\delta\beta} + \frac{\delta\epsilon}{3\beta}\right)^{-1+o(1)} \quad \text{and} \quad P_g = L_p\left(1/3, \frac{1}{3\delta\beta}\right)^{-1+o(1)}.$$

Equation (*) then becomes

$$2\epsilon - \frac{2}{3\delta\beta} - \frac{\epsilon\delta}{3\beta} = \beta. \tag{7.3}$$

Equation (**) also translates into

$$2\beta = \beta + \frac{1}{3\beta\delta} + \frac{\epsilon\delta}{3\beta}. \tag{7.4}$$

If one computes $2 \cdot (7.3) - (7.4)$, one obtains:

$$\epsilon = \frac{9\beta^2}{6\beta + \delta}. \tag{7.5}$$

When injecting the value of ϵ in Equation (7.4) we obtain a degree 2 equation of unknown δ for which β is a parameter:

$$\delta^2(6\beta^2) + \delta(1 - 18\beta^3) + 6\beta = 0. \tag{7.6}$$

The equation has negative real solutions for $0 < \beta \leq ((5-2\sqrt{6})/18)^{1/3}$ and positive real solutions for $\beta \geq ((5+2\sqrt{6})/18)^{1/3}$. We set β to its minimal value, compute $\delta = \beta(3\sqrt{6}-6)$ from Equation (7.6) and $\epsilon = \beta\sqrt{6}/2$ from Equation (7.5). In short we find:

$$\beta \approx 0.8193, \quad \gamma \approx 0.8193, \quad \epsilon \approx 1.0034, \quad \delta \approx 1.1048. \tag{7.7}$$

Complexities and consequences The time of the linear algebra and the sieve is then $L_p(1/3, 2\beta)$ and therefore

$$\text{time(DL factory)} = L_p\left(1/3, 1.639\ldots\right)^{1+o(1)}. \tag{7.8}$$

The preparation of the permanent file has a cost $L_p(1/3, 2\epsilon)$, which implies

$$\text{time(file preparation)} = L_p(1/3, 2.007\ldots)^{1+o(1)}. \tag{7.9}$$

The size of the permanent file is $E^2 P_g$, which equals the time for the sieving, up to a $1 + o(1)$ exponent:

$$\text{space(DL factory)} = L_p(1/3, 1.639\ldots)^{1+o(1)}. \tag{7.10}$$

Note first that one can reduce the time of the file preparation by increasing the time of the DL factory. Note next that the space complexity is much larger than that of classical NFS, i.e., $L_p(1/3, 0.96\ldots)^{1+o(1)}$.

A consequence of optimizing the parameters for the DL factory is the decrease of the smoothness parameter B, with consequences on the individual logarithm stage.

7.3 Complexity of descent steps

The *individual logarithm* stage, as introduced in Section 6.3.5, is done in two stages: smoothing and descent. We analyzed the smoothing in Chapter 4 and found a complexity of $L_p(1/3, 1.232\ldots)^{1+o(1)}$. It imposes that the size of the largest special-Q to be descended is $L_p(2/3, c)$ with $c = 0.811\ldots$. Let us now focus on the descent.

Let λ be a positive parameter less than 1 and let E' be a so-called sieve parameter in special-Q. Let $\mathfrak{q}$ be a degree 1 prime ideal of norm q in the number field of f and let (a_1, b_1) and (a_2, b_2) be two vectors of $\mathrm{Lat}(\mathfrak{q})$ which form a basis. Hence we list the pairs (a, b) of $\mathrm{Lat}(\mathfrak{q})$ by considering pairs (i, j) in $[0, E'] \times [-E', E']$ and putting $a = ia_1 + ja_2$ and $b = ib_1 + jb_1$. We also put $F_\mathfrak{q} = F(a_1 i + a_2 j, b_1 i + b_2 j)$ and $G_\mathfrak{q} = G(a_1 i + a_2 j, b_1 i + b_2 j)$. Then, a descent step is nothing more than searching a pair (i, j) in $[0, E'] \times [-E', E']$ such that

- $F_\mathfrak{q}(i, j)/q$ is q^λ-smooth;
- $G_\mathfrak{q}(i, j)$ is q^λ-smooth,

or the algorithm obtained when the roles of f and g are exchanged.

We have shown in Section 6.3.5 that the complexity of descent is $T^{1+o(1)}$ where T is the complexity of a descent step. Let us evaluate T depending on the values of β and δ chosen in Section 7.1 and respectively Section 7.2. It will allow us to verify that in the classical variant of NFS and in the DL factory, the individual logarithm stage is negligible with respect to the main phase.

We have to set parameter λ, for which we choose a constant value for ideals $\mathfrak{q}$ of large and intermediate norm and a different one for ideals of small norm, as explained below. We also have the choice of a positive constant ϵ' such that $E' = L_p(1/3, \epsilon')$. Similarly ϵ' has a value for large and medium ideals and a second one for small ideals.

We split the prime ideals $\mathfrak{q}$ to be descended in three types depending on their size $q := \mathrm{N}(\mathfrak{q})$:

- (large) $q = L_p(2/3, c)$ with $c > 0$; we use ECM to test the q^λ-smoothness of all the pairs (i, j) in $[0, E'] \times [-E', E']$. Note that the q^λ-smoothness tests with ECM take a time $L_p(1/3, \cdot)$.
- (medium) $q = L_p(\gamma, \cdot)$ with $1/3 < \gamma < 2/3$; we proceed as in the large case, with a time for ECM of type $L_p(\gamma/2, \cdot)$ negligible with respect to $L_p(1/3, \cdot)$.
- (small) $q = L_p(1/3, c)$ with $c > 0$; for practical reasons here we sieve on $[0, E'] \times [-E', E']$ using the special-Q technique that we present in Algorithm 8.1.

Remark that, for large ideals $\mathfrak{q}$, we cannot rely exclusively on a sieving technique. Indeed, the number of primes to be sieved, i.e., less than q^λ, is $L_p(2/3, \cdot)$. However, sieving might accelerate the computations if it allows to quickly select good candidates for the ECM tests (see [JL03]).

We impose that (a_1, b_1) and (a_2, b_2) have coefficients of size approximatively $\sqrt{q}$, as can be obtained due to Section 8.1.2. Then, for all $|i|, |j| \leq E'$ we have

$$\begin{array}{rcl} \left|F_{\mathfrak{q}}(i,j)\right| & \leq & (d+1)E'^d q^{d/2} p^{1/d}, \\ \left|G_{\mathfrak{q}}(i,j)\right| & \leq & 2E' q^{1/2} p^{1/d}. \end{array} \tag{7.11}$$

Furthermore, we make the heuristic assumption that for random integers i and j, the integers $G_{\mathfrak{q}}(i,j)$ and $F_{\mathfrak{q}}(i,j)/q$ are q^λ-smooth with the same probability as random integers of the same size. We next make the heuristic assumption that the smoothness probabilities of the two sides are independent.

The probability that the two norms are q^λ-smooth is greater or equal to the probability that a number of the same size as the product of the norms is q^λ-smooth. Therefore the probability that a pair (i,j) descends, i.e., $F_{\mathfrak{q}}(i,j)/q$ and $G_{\mathfrak{q}}(i,j)$ are q^λ-smooth, is as follows

$$\text{Prob(a pair } (i,j) \text{ descends)} \geq \rho\left(\frac{(d-1)(\log q)/2 + (d+1)\log E' + (2\log p)/d}{\lambda \log q}\right)^{1+o(1)} \tag{7.12}$$

$$= \rho\left(\frac{d-1}{2\lambda} + \frac{(d+1)\log E'}{\lambda \log q} + \frac{2\log p}{d\lambda \log q}\right)^{1+o(1)}. \tag{7.13}$$

In the case of large q, the argument of $\rho()$ in Equation (7.13) is $(1+o(1))\frac{d}{2\lambda}$. Hence, the inverse of the q^λ-smoothness probability multiplied by the cost of a q^λ-smoothness test is as follows

$$\text{time(descent large } q) = L_p\left(1/3, \frac{\delta}{6\lambda} + 2\sqrt{\frac{c\lambda}{3}}\right)^{1+o(1)}. \tag{7.14}$$

We minimize the time by taking

$$\lambda = \left(\frac{\delta^2}{12c}\right)^{1/3}. \tag{7.15}$$

Note that this is less than 1 when δ and c belong to the interval $[\frac{3}{4}, \frac{3}{2}]$. We obtain a complexity as follows

$$\text{time (descend large } q) = L_p\left(1/3, \left(\frac{3\delta c}{2}\right)^{1/3}\right)^{1+o(1)}, \tag{7.16}$$

where $c \approx 0.811$ was imposed by the smoothing stage and where $\delta \approx 1.44$ or 1.10 depends on the variant of NFS–classical or DL factory.

The case of medium q is the fastest. Indeed, the argument of ρ in Equation (7.13) is $(1+o(1))\frac{d}{2\lambda}$. The complexity of ECM with a smoothness bound of size less than q is negligible compared to $L_p(1/3, \gamma)$ for any $\gamma > 0$. Hence the complexity is $L_p(1/3, \frac{\delta}{6\lambda})$, which is smaller than that of descending a large $\mathfrak{q}$.

Since the medium case is very fast, we can impose a stronger smoothness condition. Indeed, with the price of slightly increasing the cost of each descent

variant	small q	large q
classical NFS	0.97	1.21
DL factory	1.19	1.10

Table 7.1: Values of the constant τ such that the complexity of a descent step is $L_p(1/3, \tau)^{1+o(1)}$.

step we can reduce the height of the descent tree. Inspired from [Cop84], we descend ideals of size $q = L_p(l_q, \cdot)$ to ideals of size $L_p(\frac{l_q+1/3}{2}, \cdot)$ if we impose a smoothness bound of $\exp\left(\sqrt{\log q \log B}\right)$. One can check that the total number of nodes in the middle part of descent tree is $\exp\left((\log\log p)(\log\log\log p)\right)$ instead of $\exp\left((\log\log p)^2\right)$.

The hardest descent step corresponds to small prime ideals $\mathfrak{q}$ of size $L_p(1/3, u)$ with $u > 0$. For fixed E', the argument of ρ in Equation (7.13) increases with q. When the smoothness probability increases, one can decrease E' and further increase the probability. Hence the hardest descent step corresponds to the case when $q^\lambda = B$. When replacing $(\lambda \log q)$ by $\log B$ in Equation (7.13), we obtain

$$\text{Prob}\left(\text{a small } q \text{ descends}\right) = L_p\left(1/3, \frac{\delta}{6\lambda} + \frac{\delta\epsilon'}{3\beta} + \frac{2}{3\delta\beta}\right)^{-1+o(1)}. \tag{7.17}$$

The optimality of E' implies that, the cardinality of the lattice points that we sieve equals the inverse of the success probability. This writes as follows

$$2\epsilon' = \frac{\delta}{6\lambda} + \frac{\delta\epsilon'}{3\beta} + \frac{2}{3\delta\beta}. \tag{7.18}$$

This shows that the optimal value for λ is 0.999, i.e., close but not equal to 1, and we have

$$\text{time}\left(\text{descend small } q\right) = L_p\left(1/3, \left(\frac{\delta}{6} + \frac{2}{3\delta\beta}\right) / \left(2 - \frac{\delta}{3\beta}\right)\right)^{1+o(1)}, \tag{7.19}$$

where β and δ are the values imposed by the main phase.

Numerical values and comments Let us write the time of descending one ideal $\mathfrak{q}$ as $T = L_p(1/3, \tau)^{1+o(1)}$. We summarize the values of the constants τ for the classical and the DL factory variants in Table 7.1. One can also remark that the most expensive descent step corresponds to large $\mathfrak{q}$s for the classical NFS and to small $\mathfrak{q}$s for the DL factory. In both cases, the descent is negligible with respect to the smoothing step, whose complexity is $L_p(1/3, 1.232\ldots)^{1+o(1)}$ (see Chapter 4). In turn, regardless on the variant, the individual logarithm stage is negligible with respect to the main phase.

7.4 Zoology of NFS variants

The large number of NFS variants might be confusing. It motivated us to make a recapitulation table of the complexities of the different versions.

The multi-field version of NFS, proposed in [Mat03] and improved in [CS06], has one rational side and multiple algebraic sides. In our Master Thesis we found that in the multi-field variant the complexity of the descent remains less than $L_p(1/3, 1.232)$. A simple argument would be to say that the probability of a successful pair (i, j) in the descent steps is increased, but we also have to consider the small change in the value of δ and β.

The special number field sieve for discrete logarithm (SNFS) was proposed in [Sem02]. Semaev proposed to set the parameters such that the smoothing step takes time $L_p(1/3, 1.446\ldots)$ and showed that it was the dominating step of the individual logarithm stage. As an alternative method we can use the same technique as for the classical variant and obtain a constant of $\sqrt[3]{3} \approx 1.442$ which is slightly better. Our improved smoothing method also applies so we obtain the new complexity of $L_p(1/3, 1.232)$. For the SNFS values of the parameters β and δ, $\delta = 3^{2/3}2^{-1/3} \approx 1.44$ and $\beta = \sqrt[3]{4/9} \approx 0.763$ we obtain that the descent has a complexity less than $L_p(1/3, 1)$, which is negligible with respect to the complexity of the smoothing step.

In Table 7.2 we write in each case a constant c so that the complexity of the corresponding stage is $L_p(1/3, c)^{1+o(1)}$. We write in bold characters the new complexities.

variant	sieve	linear algebra	individual logarithm	file preparation
classical NFS	1.923	1.923	1.442/**1.232**	none
SNFS	1.526	1.526	1.461/**1.232**	none
multi-field NFS	1.902	1.902	1.442/**1.232**	none
DL factory	**1.639**	**1.639**	**1.232**	**2.007**

Table 7.2: Complexity of the NFS variants of discrete logarithm. For any complexity $L_p(1/3, c)^{1+o(1)}$ we write a three decimal approximation of c.

7.5 The function field sieve

Let us analyze the classical variant of the FFS algorithm when g is a linear polynomial and f is monic with small coefficients. We call θ the root of f in its function field. The parameters to be set are

- d = the degree of f in x;
- e = the sieve parameter, i.e., the pairs (a, b) considered in the sieve stage are polynomials in $\mathbb{F}_q[t]$, a monic, of degree at most e, which corresponds to the D parameter in [JL06];
- b_1 = the smoothness bound, i.e., the factor base is formed of irreducible polynomials of degree up to b_1 and of prime ideals $\langle q_1, \theta - \rho\rangle$ with $\deg q_1 \leq b_1$.

We distinguish the general case and the middle prime case according to the size of q when compared to q^k. Note that both the classical FFS and the two rational side FFS apply in the general and the middle prime case.

General case: $q = L_{q^k}(\alpha, c)$ with $0 \leq \alpha < 1/3$ and $c > 0$. Before any other computation we find the degree of g in t. Since $\mathrm{Res}_x(f, g)$ must have a degree k divisor, we have $\deg \mathrm{Res}_x(f, g) \geq k$. Since this degree is at most $\deg_t g \cdot d + \deg_t f$ and f has small degree in t we obtain $\deg_t g \approx k/d$.

Let P_f and P_g be the probabilities that $F(a, b)$ and, respectively $a - bm$, are b_1-smooth when (a, b) is a pair of the sieving domain. The number of pairs found by the sieve equals $P_f P_g q^{2e-1}$ and must slightly exceed the cardinality of the factor base, which is less than $2q^{b_1}$. Hence we impose

$$P_f P_g q^{2e-1} = 2q^{b_1(1+o(1))}. \tag{7.20}$$

Next, the time of the sieve is q^{2e-1} and the time of the linear algebra stage is $(q^{b_1})^{2+o(1)}$. As in the NFS algorithm, we balance the two stages by imposing

$$e = b_1. \tag{7.21}$$

By analogy with the NFS algorithm, we expect the cardinality of the factor base to be $L_{q^k}(1/3, \cdot)$. Hence, we impose

$$e = b_1 = \frac{\beta k^{1/3} (\log k)^{2/3}}{\log q^{2/3}}, \tag{7.22}$$

for a constant $\beta > 0$ to be computed. As we argue in Remark 9.2.2, it is optimal to have the average degrees of $F(a, b)$ and $a - bm$ of comparable size. We have $\deg_t \big(F(a, b)\big) \leq de + \deg_t f \approx de$, because f can have coefficients in t as small as desired. We also have $\deg(a - bm) \approx k/d + e$. Therefore we impose $(d - 1)e = \text{constant} \cdot k/d$ which implies that d must be of type

$$d = \delta \left(\frac{k \log q}{\log k} \right)^{1/3}, \tag{7.23}$$

where δ is a constant. By Theorem 1.3.1, we have $P_f = u^{-u(1+o(1))}$ where $u = \frac{de}{b_1} = d$, so $P_f = L_{q^k}(1/3, \frac{\delta}{3})^{-1+o(1)}$; we used $b_1 = e$. Similarly $P_g = v^{-v(1+o(1))}$ where $v = \frac{k/d}{b_1}$ and therefore $P_g = L_{q^k}(1/3, \frac{1}{3\delta\beta})^{-1+o(1)}$. When injecting in Equations (7.20) and (7.21) we obtain

$$\frac{1}{3\delta\beta} + \frac{\delta}{3} = \beta. \tag{7.24}$$

We minimize the value of β–and therefore the computing time–such that there exists a positive value of δ satisfying the above equation. It is a degree 2 equation with discriminant $\beta^2 - \frac{4}{9\beta}$. Hence we take, $\beta = \sqrt[3]{4/9}$, which implies $\delta = \frac{3}{2}\beta > 0$. Since the computing time is $(q^{b_1})^{2+o(1)}$ we obtain the heuristic complexity of the main phase of FFS:

$$\text{time (FFS)} = L_{q^k}\left(1/3, \sqrt[3]{\frac{32}{9}}\right). \tag{7.25}$$

Remark 7.5.1. We underline the similarity of FFS with SNFS, as analyzed by Semaev [Sem02]. Indeed, in both cases, the coefficients of the polynomial f are small and in both cases the bit-size of the coefficients of g equals $1/d$ times the bit-size of the cardinality of the finite field. If in SNFS we set $B = \beta(\log p)^{1/3}(\log\log p)^{2/3}$ and $d = \delta(\log p)^{1/3}(\log\log p)^{-1/3}$, then Equation (7.24) holds for both FFS and SNFS. In short one can say that "The function field sieve is quite special".

The middle prime case: $q = L_{q^k}(1/3, \alpha)$ with $\alpha > 0$. Note that in Equation (7.20) we assumed that we can tune parameters e and b_1 so that the sieving domain contains just enough relations so that we obtain a square matrix for the linear algebra. In practice, when q increases with respect to q^k, this assumption is increasingly hard to meet. From an asymptotic point of view, when q is in the middle prime case, since the cardinality of the factor base is q^{b_1}, the complexity of the linear algebra is $L_{q^k}(1/3, 2\alpha b_1)^{1+o(1)}$. In particular, we cannot impose that none of the two stages of the main phase, the sieve and the linear algebra, is negligible with respect to the other. But let us present the analysis in [JL06], which will allow us to compare FFS with the new algorithm in [Jou13a].

For each constant $D \geq 1$ in $\mathbb{N}$ we define an algorithm which consists in running the classical variant of FFS with parameters $e = D$ and $b_1 = D$. We will obtain the same complexities as for the two rational side variant in [JL06]. Let us determine, for each constant $\alpha > 0$, which is the best algorithm for the finite fields $\mathbb{F}_{q^k}$ with $q = L_{q^k}(1/3, \alpha)$ and what is the complexity of the algorithm when q goes to infinity. Note that we have

$$k = \frac{1}{\alpha}\left(\frac{\log Q}{\log\log Q}\right)^{1/3}, \tag{7.26}$$

where $Q = q^k$.

The cardinality of the pairs (a, b) in the sieving domain is q^{2D-1}: there are q^D values for b and q^{D-1} values for a as we imposed that a is monic. Also, the cardinality of the factor base is bounded by $2q^D$, so the complexity of the linear algebra stage is $L(1/3, c_2)^{1+o(1)}$ with

$$c_2 = 2\alpha D. \tag{7.27}$$

The time of the sieve is $P_f^{-1}P_g^{-1}2q^D$ where P_f and P_g are the smoothness probabilities of the f and the g side respectively. Note however that this is not necessarily equal to the cardinality of the sieving domain as it was for the general case. Since $\sqrt{k/D}$ goes to infinity with q, we can tune $\deg_x f$ to approximatively $\sqrt{k/D}$ and therefore we can have $\deg F(a, b) \approx \deg(a - bm) \approx \sqrt{kD}$. Hence, $P_f = P_g = L_Q\left(1/3, \frac{1}{\sqrt{3\alpha D}}\right)^{-1+o(1)}$. The cost of the sieve is then $L_Q(1/3, c_1)^{1+o(1)}$ with

$$c_1 = D + \frac{2}{3\sqrt{\alpha D}}. \tag{7.28}$$

One can check that the condition that the sieving domain contains enough coprime pairs is

$$(D + 1)\alpha \geq \frac{2}{3\sqrt{\alpha D}}. \tag{7.29}$$

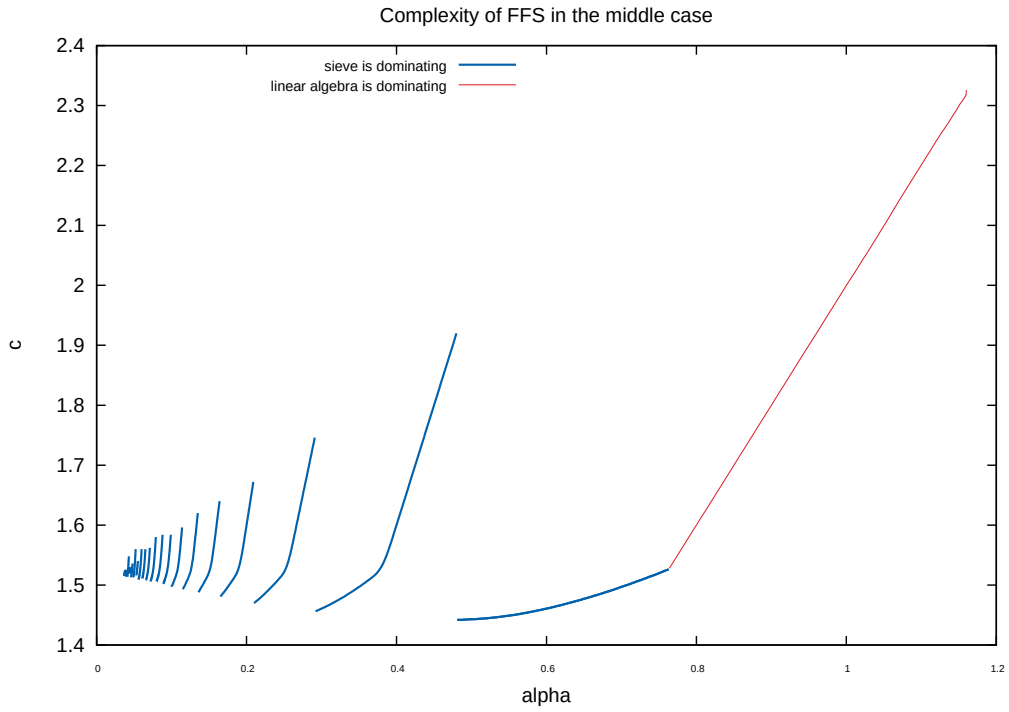


Figure 7.1: Complexity of FFS in the middle prime case. We plot c such that the complexity equals $L_Q(1/3, c)^{1+o(1)}$ with respect to α so that $q = L_Q(1/3, \alpha)^{1+o(1)}$. We distinguish the ranges where the sieve and the linear algebra stage dominate each other.

In order to visualize the complexity, we considered every value of α in the interval $[0.001, 2]$. For each α we tested the two values of D closest to the value of D which would equalize the time of the sieve and of the linear algebra. Hence we obtained Figure 7.1. Of course, we switch to NFS-HD (see [JLSV06]) when the complexity is higher than $L_Q\left(1/3, \sqrt[3]{\frac{128}{9}}\right)$ (see [JLSV06]).

Remark 7.5.2. Joux [Jou13a] proposed a new algorithm for the same range of fields. This new algorithm, uses a faster technique for the relation collection, called *pinpointing*. In the same time, this new algorithm uses the same factor base as FFS. Hence, FFS and the new algorithm take the same asymptotic time for the range where the linear algebra dominates the relation collection stage. On Figure 7.1 one sees that the sieve dominates for small values of α while the linear algebra dominates when α is large. Joux showed that the crossing point between the sieve and linear algebra dominance is $(2/3)^{2/3}$. Hence, FFS is as fast as the pinpointing algorithm for fields $\mathbb{F}_{q^k}$ where $q = L_{q^k}(1/3, \alpha)^{1+o(1)}$ with $\alpha \geq (2/3)^{2/3}$.

Chapter 8

Classical improvements and variations on NFS and FFS

Now that we made the complexity analysis of NFS and FFS, we can focus on the improvements which do not change the complexity but have a major impact in practice. Indeed, an improvement which is hidden in the $o(1)$ of the exponent can mean a factor 20 in practice and hence make the difference between being feasible and not. We do not seek exhaustiveness but rather present the two most important improvements (the lattice sieve and block Wiedemann). We also deal with two improvements which were not well explained in the literature (dealing with two non-linear polynomials, selecting polynomials for NFS in the non-prime case).
We start the chapter with the sieving technique first used in the factorization context, which is also used in the descent. Then we show how to parallelize and distribute the computations in the linear algebra stage. We show how to select polynomials for fields $\mathbb{F}_{p^n}$ with $n > 1$. We conclude with the study of the technicality which occurs in the smoothing step of the individual logarithm stage when both polynomials are non-linear.

8.1 The lattice sieve

8.1.1 The lattice sieve technique

Pollard [Pol93] proposed a sieving technique, called lattice sieve, which is faster than the line sieve, that we presented in Section 6.3.2. It can be used in numerous sieving algorithms, in particular for the FFS algorithm and the NFS algorithm, in both its factorization and discrete logarithm versions. The basic principle is to sieve only the pairs (a, b) in the sieving domain whose norms are divisible by a prime q smaller than the smoothness bound. After we introduce the algorithm which allows to sieve pairs (a, b) of this type, we present in detail one of its subroutines which finds short vectors in a 2-dimensional lattice. Then we conclude the section by analysing the speed-up between the lattice and the line sieve.

Let us introduce the special-Q technique. Let α_f and α_g be roots of f and g

in their associated fields. Let $\mathfrak{q} = \langle q, \alpha_f - \rho \rangle$ be a prime ideal in the field defined by f. Of course, any result in this section is true if one exchanges the roles of f and g (it happens often in the implementations of the SNFS). In the individual logarithm stage, one uses the special-Q technique for ideals on both the f and g side. In the sieve stage we can use for simplicity the f side only, but in practice it can be interesting to also use the g side. We introduce two new parameters: B' which is the smoothness bound of the special-Q technique and E', which is the sieve parameter for the special-Q procedure. When analysing the complexity of the individual logarithm stage (see section 7.3), we chose the value of E' and we set $B' := q^\lambda$ for a positive constant λ. In the lattice sieve, we set B' to B and E' to $E/\sqrt{q}$; this choice will be explained later. It is time to present the special-Q technique in Algorithm 8.1.

Algorithm 8.1 The special-Q technique.

Input: a prime ideal $\mathfrak{q} = \langle q, \alpha_f - \rho \rangle$ and two parameters: $E' > 0$ and $B' > 0$

Output: all the coprime pairs (a, b) such that $a - b\alpha_f \equiv 0 \bmod \mathfrak{q}$ and

$$\mathrm{N}(a - b\alpha_f)/q \text{ and } \mathrm{N}(a - b\alpha_g) \text{ are } B'\text{-smooth.}$$

1: Find two vectors $v_1 = (a_1, b_1)$ and $v_2 = (a_2, b_2)$ on $\mathrm{Lat}(\mathfrak{q})$ with $\|v_i\|_\infty$ of order $\sqrt{q}$ for $i = 1, 2$

2: $F_\mathfrak{q}(X, Y) \leftarrow F(a_1X + a_2Y, b_1X + b_2Y)$

3: $G_\mathfrak{q}(X, Y) \leftarrow G(a_1X + a_2Y, b_1X + b_2Y)$

4: Sieve the coprime pairs (i, j) in $[0, E'] \times [-E', E']$ such that

$$F_\mathfrak{q}(i, j)/q \text{ and } G_\mathfrak{q}(i, j) \text{ are } B'\text{-smooth}$$

5: For each pair (i, j) found at the previous step, output $(a, b) := (a_1i + a_2j, b_1i + b_2j)$

A first question is why the special-Q technique yields coprime pairs.

Lemma 8.1.1. *If the pair (i, j) is coprime and parameter $E' < \sqrt{q}$, then the corresponding pair (a, b) is coprime.*

Proof. One has

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a_1 & a_2 \\ b_1 & b_2 \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix}. \tag{8.1}$$

Since the 2×2 matrix above has determinant q, the unique divisor that a and b can have in common is q. Since $E' < \sqrt{q}$ and the vectors (a_1, b_1) and (a_2, b_2) have norms of size at most $\sqrt{q}$, we have $\max(|a|, |b|) < q$; hence (a, b) is a coprime pair. □

Before going on with the questions related to the special-Q technique we note that the sieve in line 4. of Algorithm 8.1 is not necessarily the line sieve. One can use the so-called sieve by vectors described in Appendix A. of [KAF+10] to deal with the primes p which are large when compared to E'. But the large number of improvements makes it interesting to have a look directly to the code of some implementations like GGNFS [K+05] and CADO-NFS [BFG+09].

8.1.2 Computing short vectors

Let $\mathfrak{q} = \langle q, \alpha_f - \rho \rangle$ be a prime ideal and consider its associated lattice $\mathrm{Lat}(\mathfrak{q})$. Let $v_1 = (a_1, b_1)$ and $v_2 = (a_2, b_2)$ be two vectors which form a basis of $\mathrm{Lat}(\mathfrak{q})$. We also put $F_\mathfrak{q}(i,j) = F(a_1 i + a_2 j, b_1 i + b_2 j)$ and $G_\mathfrak{q} = G(a_1 i + a_2 j, b_1 i + b_2 j)$. One has

$$\forall (i,j) \in [-E', E']^2, \quad \left|F_\mathfrak{q}(i,j)\right| \leq \|f\|_\infty \max\left(\|v_1\|_\infty, \|v_2\|_\infty\right)^{\deg f} (\deg f + 1) E'^{\deg(f)}. \tag{8.2}$$

A similar equation holds for g; for example if g is linear we have, for all pairs (i,j), $|G_\mathfrak{q}(i,j)| \leq 2\|g\|_\infty \max\left(\|v_1\|_\infty, \|v_2\|_\infty\right) E'$. Hence, in order to increase the smoothness probability it is important to reduce $\max(\|v_1\|_\infty, \|v_2\|_\infty)$.

When given a basis of a 2-dimensional lattice, Algorithm 8.2 computes a basis formed of short vectors.

Algorithm 8.2 Lattice basis reduction in dimension 2.

Input: Two vectors u and v in $\mathbb{Z}^2$
Output: Two vectors u' and v' such that $\langle u', v' \rangle \leq \frac{1}{2}\|u'\|_2 \|v'\|_2$

```
while ⟨u, v⟩ > (1/2)‖u‖_2 ‖v‖_2 do
    if ‖u‖_2 > ‖v‖_2 then
        exchange u and v
    end if
    μ* ← ⟨u, v⟩ / ‖u‖_2^2
    μ ← ⌊μ*⌉
    v ← v − μu
end while
output u and v
```

The complexity of Algorithm 8.2 is that of the extended Euclidean algorithm, as proven in Chapter 16 in [vzGG03].

The output of the algorithm is not necessarily the pair (u, v) which optimizes $\max(\|u\|_\infty, \|v\|_\infty)$ but in practice it is close to that. Indeed, one of u and v has a small Euclidean norm, as shown by the next classical result, which generalizes to lattices of any dimension.

Proposition 8.1.2 (Chapter 16,[vzGG03]). *Let u and v be two vectors of $\mathbb{Z}^2$ and call q their determinant. If $\langle u, v \rangle \leq \frac{1}{2}\|u\|_2 \|v\|_2$, then one has*

$$\min\left(\|u\|_2, \|v\|_2\right) \leq \left(\frac{4}{3}\right)^{1/4} \sqrt{q}. \tag{8.3}$$

Let now u and v be the vectors obtained in the NFS algorithm when reducing $\mathrm{Lat}(\mathfrak{q})$ for a prime ideal $\mathfrak{q}$. Since $\det(u, v)$ equals the determinant of $\mathrm{Lat}(\mathfrak{q})$, one has $u_1 v_2 - u_2 v_1 = q$. Experiments show that, for almost all ideals $\mathfrak{q}$, the short vectors u and v have roughly equal norm. Hence $\|u\|_\infty$ and $\|v\|_\infty$ have a size of approximatively $\sqrt{q}$. For the relatively few prime ideals $\mathfrak{q}$ for which the reduced basis is unbalanced, i.e., there is a large ratio between the lengths of the output

vectors, in NFS we skip $\mathfrak{q}$ as it is more efficient to sieve on lattices of larger prime ideals rather than on an unbalanced lattice.

Algorithm 8.2 was translated to FFS in [JL02]. In this case, the algorithm and the analysis are simplified as the scalar μ is replaced by the Euclidean quotient and the absolute value of an integer is replaced by the degree of a polynomial.

8.1.3 Evaluating the speed-up

Pollard [Pol93] gave brief arguments for the efficiency of the lattice sieve. We show here that, under a plausible assumption, the lattice sieve is $\Theta(\log B)$ times faster than the line sieve.

Let B be the smoothness parameter and put $B_1 = cB$ for a constant $c < 1$. The lattice sieve algorithm consists in using the special-Q technique for all the prime ideals $\mathfrak{q}$ in the field of f with norms in the interval $[B_1, B]$. Parameter E' is chosen equal to $E/\sqrt{q}$ so that the pairs (a, b) obtained have the same size as in the line sieve. Indeed, we saw that heuristically one can find two vectors in $\mathrm{Lat}(\mathfrak{q})$ whose norm has size $\sqrt{q}$. Then, $E' = E/\sqrt{q}$, guarantees that the special-Q technique considers the pairs (a, b) in the intersection of $[0, E] \times [-E, E]$ with $\mathrm{Lat}(\mathfrak{q})$.

The following result shows that in the lattice sieve one has a smaller sieving cost, whereas only a small fraction of the doubly-B-smooth coprime pairs are missed. For simplicity we state it in the NFS context, but it also applies for the FFS algorithm.

Proposition 8.1.3. *Let Q be the cardinality of a finite field. Call E the sieve parameter in the line sieve algorithm and B the smoothness bound in the NFS algorithm. We use the property that $B = L_Q(1/3, \beta)^{1+o(1)}$ and $E = L_Q(1/3, \epsilon)^{1+o(1)}$ for two constants β and ϵ. Then the following assertions are true.*

1. *The cost of the lattice sieve when $B_1 = cB$, for a constant $c < 1$, is $\frac{\log cB}{|\log c|}$ times smaller than that of the line sieve.*

2. *Let us make the hypothesis that, for any prime ideal $\mathfrak{q}$, the norm of a pair in Lat($\mathfrak{q}$) has the same smoothness probability as a random number of the same size. Then the number of doubly-B-smooth pairs missed by the lattice sieve equals $o(1)$ of the total when Q goes to infinity.*

Proof. 1. Let S equal $2E^2$, the cardinality of the sieving domain. Then, the cost of the line sieve is

$$W_{\text{line}}(S, B) = \sum_{\ell \leq B \text{ prime}} \left(\log(\ell)^{O(1)} + \frac{S}{\ell} \right). \tag{8.4}$$

For any constant $k \in \mathbb{N}^*$, the sum $\sum_{\ell \leq B, \text{ prime}} (\log \ell)^{k+1}$ is smaller than $B(\log B)^k$ which is negligible with respect to S, which is roughly B^2. Hence we have

$$W_{\text{line}}(S, B) = \big(1 + o(1)\big) \left(\sum_{\ell \leq B \text{ prime}} \frac{1}{\ell} \right) S. \tag{8.5}$$

In the lattice sieve, for each prime ideal $\mathfrak{q} = \langle q, \alpha_f - \rho \rangle$ with $q \in [B_1, B]$, one does the work corresponding to the line sieve with sieve parameter $E' = E/\sqrt{q}$; hence to a sieve domain of S/q pairs. Then, the total cost of the lattice sieve is

$$W_{\text{lattice}}(S, B) = \sum_{q \in [B_1, B], \text{prime}} W_{\text{line}}(S/q, B) \tag{8.6}$$

$$= (1 + o(1))S \left(\sum_{q \in [B_1, B]} \frac{1}{q} \right) \left(\sum_{\ell \leq B \text{ prime}} \frac{1}{\ell} \right). \tag{8.7}$$

The fraction of work done in the lattice sieve with respect to the line sieve is then $\sum_{q \in [B_1, B]} \frac{1}{q}$. We use the formula

$$\sum_{p \text{ prime} \leq X} \frac{1}{p} = \log\log X + M + o(1), \tag{8.8}$$

where M is the Meissel-Mertens constant. We obtain

$$\begin{aligned} \frac{\text{cost(lattice sieve)}}{\text{cost(line sieve)}} &= \log\log B - \log\log cB \\ &= \log\left(1 + \frac{|\log c|}{\log cB}\right) \\ &= \big(1 + o(1)\big) \frac{|\log c|}{\log cB}. \end{aligned}$$

2. The pairs (a, b) missed when using the lattice sieve are those whose norm on the f side has no prime factor in the interval $[B_1, B]$. We have then to show that we have

$$\frac{\psi(x, cB)}{\psi(x, B)} = o(1), \tag{8.9}$$

where x is the size of the norms on the f side. For this we use the following formula

$$\psi(x, B) \sim \prod_{p \leq B} \frac{\log x}{p \log p}. \tag{8.10}$$

Then we have $\psi(x, cB)/\psi(x, B) = \prod_{p \in [cB, B]} \frac{p \log p}{\log x}$. Since $B = L_Q(1/3, \cdot)$ and $x = L_Q(2/3, \cdot)$, for large enough Q we have that each factor of the product is less than $1/2$. Hence, the fraction of lost pairs goes to 0 when Q goes to infinity. □

The expression $\log\log B - \log\log cB$ might be not intuitive. A numerical example is as follows

$$\sum_{q \in [2^{20}, 2^{21}], \text{ prime}} \frac{1}{q} \approx 0.049. \tag{8.11}$$

Hence one can use the following rule of thumb: the speed-up of the lattice sieve with respect to the line sieve is the bit-size of the smoothness bound B.[1]

[1] As an anecdote, the experiment made by Pollard to show the effectiveness of the lattice sieve was 30% slower than the line sieve [Pol93].

Large prime variation Note though that the analysis is more complex when considering the large prime variation. This latter goes as follows. We consider a new parameter B_1 smaller than B, called the factor base bound. For each prime q in a subset of the interval $[B_1, B]$, called special-Qs, we do a line sieve procedure for primes ℓ up to B_1. Then we run a smoothness test with the smoothness bound B to all pairs whose norm, after eliminating the factors smaller than B_1, is less than a threshold τ. In this new context, the smoothness bound B is renamed large prime bound. In short, when using the large prime variation in NFS, by sieving one collects pairs whose norms can be written as follows

$$\text{norm} = \prod_{\ell \leq B_1} \ell^{v(\ell)} \cdot q \cdot \prod_{L \in [B_1, B]} L^{v(L)}, \tag{8.12}$$

for a prime q in $[B_1, B]$ and where the product of the large primes $L \in [B_1, B]$ is less than the threshold τ.

Since, in the NFS algorithm, we end the sieving when enough pairs are collected, not all the primes q in the interval $[B_1, B]$ are used as a special-Q. A pair can be reported more than once, being called a duplicate, if and only if its norm has at least two prime factors which were used as a special-Q. It is then difficult to estimate the average number of times that a pair is reported. The task is even harder in practice where it seems that the number of duplicates must be kept low. For example, the record computations of [KAF+10] and [BBD+13], using the NFS and the FFS algorithms respectively, obtained an average of 1.27 and 1.42 reports per unique relation. Finally, since ECM is probabilistic, it is hard to decide if a pair is a duplicate before the end of the relation collection stage.

8.2 Parallelization of the linear algebra stage

8.2.1 General considerations on the parallelism

We presented Wiedemann's algorithm in Section 6.3.4. One disposes of two manners to share computations. The first is often called parallelization and allows to split memory. The second is called distribution and allows to share the computation time with no overhead due to the communications.

The parallelization consists in splitting the matrix A in blocks and in computing the matrix×vector product as a block product. It allows each computational unit, GPU or CPU, to store a unique block of the matrix. The drawback of this kind of sharing computations is that, when the number of computing units becomes large, the time of communications and synchronizations becomes considerable. Parallelization is used for example in the CADO-NFS softwear [BFG+09].

The distribution is obtained due to an algorithm called block Wiedemann [Cop94] that makes the object of the following subsection (also implemented in [BFG+09]).

8.2.2 Block Wiedemann

We call K and N the field and the size of the matrix B given as input to the linear algebra stage.

The most expensive step in Wiedemann's algorithm is the computation of the first $2N$ terms of the sequence yB^ix with $i = 1, 2, \ldots$ The goal of the block version of the algorithm is to replace the computation of this sequence with the computation, for a finite set of vectors x_ν, $\nu \in [1, n]$, of the sequences yB^ix_ν. This allows to distribute the computations with zero communication overhead.

Let us explain the algorithm in an informal manner; the reader can find a detailed description in [Tho03]. For a parameter n and a set of n random row vectors x_ν in $K^{N\times 1}$, consider the set

$$\left\{B^i x_\nu \mid \text{ for } i \in \left[1, \lceil N/n \rceil\right] \text{ and } \nu \in [1, n]\right\}. \tag{8.13}$$

Since the family has more than N elements, one can find a non-trivial linear combination:

$$0 = \sum_{i\in\left[1,\lceil N/n\rceil\right],\nu\in[1,n]} \lambda_{\lceil N/n\rceil - i,\nu} B^i x_\nu. \tag{8.14}$$

We set

$$w = \sum_{i\in\left[1,\lceil N/n\rceil\right],\nu\in[1,n]} \lambda_{\lceil N/n\rceil - i,\nu} B^{i-1} x_\nu. \tag{8.15}$$

Then $Bw = 0$ and $w \neq 0$ except if one could have chosen a linear combination using lower powers of B. Hence, we reduced the computation of a non-trivial solution of Bw to the computation of such a linear combination.

We call matrix linear generator of a sequence of matrices a_i with $i \in \mathbb{N}$, a linear relation of a finite number k of terms, which expresses each column of a_i as a linear combination of the columns of a_{i-1}, a_{i-2}, ..., a_{i-k}. For example, the sequence below

$$\begin{pmatrix} 1 & 1 \\ 8 & 13 \\ 55 & 89 \end{pmatrix}, \begin{pmatrix} 1 & 2 \\ 13 & 21 \\ 89 & 144 \end{pmatrix}, \ldots, \begin{pmatrix} F_i & F_{i+1} \\ F_{i+5} & F_{i+6} \\ F_{i+9} & F_{i+10} \end{pmatrix}, \ldots \tag{8.16}$$

with $i \in \mathbb{N}$, F_i being the ith Fibonacci number, admits the matrix linear generator of degree 1 in the following equation

$$a_i = a_{i-1} \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}. \tag{8.17}$$

Note that one can determine the matrix linear generator above from the first row only. Similarly, when the sequence a_i is obtained as the successive values of B^ix for a matrix B and vector x as in block Wiedemann, one can determine the matrix linear generator using only a small number m of rows of a_i. More formally, let m be a parameter and y_μ with $\mu \in [1, m]$ be a set of columns in $K^{1\times N}$. One can show that, with a probability of $1 - O(1/\#K)$, the set $\{y_\mu B^i \mid i \in$

$\left[1, \lceil N/m \rceil\right], \mu \in [1, m]\}$ spans K^N. Hence, if a vector has a zero product with all the elements of the set, then it is zero. It is therefore sufficient to find a matrix linear generator satisfied by m random rows of the matrices in Equation (8.13) for N/m consecutive values of i. We conclude that one can determine the linear generator for B using $y_\mu B^i x_\nu$ for $i \in [1, N/n + N/m]$, $\nu \in [1, n]$ and $\mu \in [1, m]$. The reader can find a review of matrix linear generator algorithms in [KY06]. One of the best results in practice corresponds to Thomé's algorithm [Tho03], with a complexity of $O\left(\frac{(m+n)^3}{mn} N (\log N)^2\right)$ operations.

Algorithm 8.3 presents the block version of Wiedemann's algorithm.

Algorithm 8.3 Block Wiedemann

Input: A singular matrix $B \in K^{N \times N}$; two parameters m and n
Output: A non-trivial solution w of $Bw = 0$

1: $x \leftarrow$Random$(K^{N \times m})$, $y \leftarrow$Random$(K^{n \times N})$
2: [Krylov] compute $a_i = yB^i x$ for i in $\left[1, \lceil N/n + N/m + m/n \rceil\right]$
3: $F \leftarrow$(matrix linear generator of the sequence a_i)
4: $C \leftarrow X^{\deg F} F(1/X)$ ▷ in $K^{n \times m}[X]$
5: Call $\sum_{i=0}^{\deg C} X^i c_i$ the first column of C, with $c_i \in K^{n \times 1}$
6: [Make Solution] compute $w \leftarrow \sum_{i=0}^{\deg C} B^i x \cdot c_i$ ▷ in $K^{N \times 1}$
7: Replace w by Bw until $Bw = 0$ and $w \neq 0$

In order to state the complexity of Algorithm 8.3, we call λ the average number of non-zero entries of the rows of B, M_0 the cost of a multiplication of an element of K by a small element and M_1 the product of two elements in K. Note that the matrix×vector product in the NFS and FFS algorithms uses operations of cost M_0 as the coefficients of the matrix B are small (except for the Schirokauer columns).

Proposition 8.2.1 (Prop. 6.12,[Tho03]). *On a field K, when using n machines, the parallel complexity of the* Krylov *and* Make solution *steps of the block Wiedemann algorithm are as follows:*

Krylov: $(\lambda M_0 + m M_1)\left(1 + \frac{n}{m}\right) N^2/n$;

Make solution: $\lambda M_0 N^2/n$.

Note that when the ratio m/n increases, the total number of matrix×vector multiplications in steps Krylov and Make solution decreases. When $m = n$ the number of matrix×vector products is $3N$, this being the case in the plain Wiedemann algorithm. When m/n is large but constant, for example 20, the number of matrix×vector products is close to $2N$.

The cost of the block version of Wiedemann's algorithm must be compared to the block Lanczos algorithm. This latter requires $2N$ matrix×vector products, but it has the drawback of a communication delay after each iteration. The reader can see an experimental comparison in [Kru11].

8.3 Polynomial selection for $\mathbb{F}_{p^n}$ with small degree n

The finite fields $\mathbb{F}_{p^n}$ with small n, possibly growing slowly with p, could not be handled with NFS more than one decade after the prime fields $\mathbb{F}_p$. Schirokauer [Sch00] proposed an algorithm when n is constant, with the same complexity as NFS in the prime case. But his method remained unimplemented due to the arithmetic difficulties when working with polynomials over number fields. A complete answer was given in [JLSV06], showing that NFS suits any finite field $\mathbb{F}_{p^n}$ when n is smaller or equal to the optimal value of parameter d, i.e., $\delta(\log Q)^{1/3}(\log\log Q)^{-1/3}$ with $Q = p^n$ (see Figure 6.1). The only changes are the polynomial selection and the fact that the two polynomials must be non-linear. This has implications on the sieve and the smoothing step of the individual logarithm stage. After seeing in this section how to handle the polynomial selection we will discuss the smoothing step in the next section.

8.3.1 The method

Let us generalize here the method proposed by Joux and Lercier for the case of prime fields [JL03], which is different than the method in [JLSV06].

We start by setting

$$d' = \lfloor d/2 \rfloor,$$

for d is the closest integer to $\delta(\log Q)^{1/3}(\log\log Q)^{-1/3}$, where $\delta = 3^{1/3}$ is the constant set in Chapter 7. We assume the slightly stronger condition that $d' \geq n+1$.

Next, we select an irreducible polynomial f in $\mathbb{Z}[x]$ which has degree $d'+1$, with as small as possible coefficients, good sieving properties and an irreducible monic factor φ of degree n modulo p. One can note the analogy with the discussion in Chapter 9.

We continue by constructing g as follows. Let us define

$$\mathrm{Lat}(\varphi, p) = \left\{ (a_0, \ldots, a_{d'}) \in \mathbb{Z}^{d'+1} \;\middle|\; \sum_{i=0}^{d'} a_i x^i \equiv 0 \mod \langle p, \varphi(x) \rangle \right\}. \tag{8.18}$$

Note that the rows of the $(d'+1) \times (d'+1)$ matrix M below form a basis of $\mathrm{Lat}(\varphi, p)$, where the first n rows have a unique non-zero entry equal to p and the next $d'+1-n$ rows are obtained from the coefficients of φ, with leading coefficient at the right.

$$M = \begin{pmatrix} p & & & \\ & \ddots & & \\ & & p & \\ & & \mathrm{vector}(\varphi) & \\ & & & \ddots \\ & & & \mathrm{vector}(\varphi) \end{pmatrix} \tag{8.19}$$

Since $\det M = Q$ and M has size $d'+1$, one finds with LLL a vector $(g_0, \ldots, g_{d'})$ of Euclidean norm approximatively $Q^{1/(d'+1)}$. We set $g = \sum_{i=0}^{d'} g_i x^i$. By construction φ is a common factor of f and g modulo p. Since φ has random coefficients modulo p, φ has Euclidean norm approximatively p, which exceeds $Q^{1/(d'+1)}$. Hence the probability that f and g share a factor in $\mathbb{Z}[x]$ is negligible.

8.3.2 Analysis

Note first that the polynomials f and g are such that $\deg f = d'+1$, $\|f\|_\infty \approx 1$, $\deg g = d'$ and $\|g\|_\infty \approx Q^{1/(d'+1)}$. As before, we denote by E the sieve parameter. One has

$$\begin{aligned} \mathrm{N}(a - b\alpha_f) &\leq E^{d'+1+o(1)} \\ \mathrm{N}(a - b\alpha_g) &\leq E^{d'+o(1)} Q^{1/(d'+1)}, \end{aligned} \tag{8.20}$$

where α_f and α_g are roots of f and g in their associated fields.

As a first approximation, the probability that a coprime pair (a, b) is doubly smooth depends on the product of the two norms of (a, b) with respect to f and g. This is

$$E^{2d'+1} Q^{1/(d'+1)} = \left(E^{d+1} Q^{2/(d+1)}\right)^{1+o(1)}.$$

Note that the right side member is the same as–or slightly smaller than–the norms product in the prime case, as computed in Equation (7.1). By taking the same parameters as in the prime case, one concludes that

$$\text{time}\left(\text{NFS in } \mathbb{F}_Q \text{ of small degree}\right) = L_Q\left(1/3, \sqrt[3]{64/9}\right)^{1+o(1)}. \tag{8.21}$$

8.3.3 Comparison with other methods of polynomial selection

Case of $n = 1$. Let us compare Joux-Lercier's method to algorithms from the factorization world. For this we fix $n = 1$, so that we address the case of prime fields, to which all the former algorithms translate easily. For a review of the polynomial selection in factoring algorithms we refer to [Bai11]. On the one hand, due to Section 6.5, f can be non-monic, so we can modify the base-m method so that $\|f\|_\infty \approx p^{1/(d+1)}$ instead of $p^{1/d}$. Indeed, we take $m = \lfloor p^{1/(d+1)} \rfloor$, so that all the coefficients of f, including the leading one, have the size of m. Other methods, especially Kleinjung's second method as described by Bai (see page 51 in [Bai11]), seem to yield polynomials with coefficients of the same average size. Hence, the product of the two norms in the factorization issued algorithms is bounded as follows

$$|F(a, b)(a - bm)| \leq E^{d+1} p^{2/(d+1)}. \tag{8.22}$$

In Table 8.1 we compare the size of the norms when $d = 4$, 5 and 6. The better properties of Joux-Lercier's method in the first approximation must be confronted to the second approximation criteria as follow. The norms of f and g are highly unbalanced in Joux-Lercier's method because the coefficients of f are almost constant, the coefficients of g are large, while $\deg f \approx \deg g$. Both Joux-Lercier and Kleinjung's algorithms can improve properties linked to the real and

		Kleinjung			Joux-Lercier		
d	d'	$\lvert a-bm\rvert$	$\lvert F(a,b)\rvert$	$\lvert a-bm\rvert\lvert F(a,b)\rvert$	$\lvert F(a,b)\rvert$	$\lvert G(a,b)\rvert$	$\lvert G(a,b)\rvert\lvert F(a,b)\rvert$
4	2	$p^{1/5}E$	$p^{1/5}E^4$	$p^{2/5}E^5$	E^3	$p^{1/3}E^2$	$p^{1/3}E^5$
5	2	$p^{1/6}E$	$p^{1/6}E^5$	$p^{1/3}E^6$	E^3	$p^{1/3}E^2$	$p^{1/3}E^5$
6	3	$p^{1/7}E$	$p^{1/7}E^6$	$p^{2/7}E^7$	E^4	$p^{1/4}E^3$	$p^{1/4}E^7$

Table 8.1: Size of the bounds on the norms of a pair (a, b) in the sieving domain, when E is the sieve parameter and d is the optimal degree of f in the base-m method of polynomial selection.

		Joux-Lercier-Smart-Vercauteren			Joux-Lercier		
d	d'	$\lvert a-bm\rvert$	$\lvert F(a,b)\rvert$	$\lvert a-bm\rvert\lvert F(a,b)\rvert$	$\lvert F(a,b)\rvert$	$\lvert G(a,b)\rvert$	$\lvert G(a,b)\rvert\lvert F(a,b)\rvert$
4	2	$p^{1/5}E^2$	$p^{1/5}E^5$	$p^{2/5}E^7$	E^3	$p^{1/3}E^2$	$p^{1/3}E^5$
5	2	$p^{1/6}E^2$	$p^{1/6}E^6$	$p^{1/3}E^8$	E^3	$p^{1/3}E^2$	$p^{1/3}E^5$
6	3	$p^{1/7}E^2$	$p^{1/7}E^7$	$p^{2/7}E^9$	E^4	$p^{1/4}E^3$	$p^{1/4}E^7$

Table 8.2: Here $n = 2$. Size of the bounds on the norms of a pair (a, b) in the sieving domain, when E is the sieve parameter and d corresponds to the optimal degree of f for prime fields.

modular roots of f and g. Indeed, in the Joux-Lercier method one can repeat the construction with different polynomials f until both f and g have good root properties.

Case of $n > 1$. Joux, Lercier, Smart and Vercauteren [JLSV06] proposed an alternative method. One starts by selecting $f_0(x)$ of degree n with small coefficients. Then one puts $f_1 = f_0(x + W)$ with W selected such that $\|f_1\|_\infty \approx 2^{d+1}p^{1/(d+1)}$. Then one reduces the $(d+1)$-dimensional lattice spanned by $f_1(x)$, ..., $x^{d-n}f_1(x)$ and p, ..., px^{n-1}. Hence one obtains a polynomial f_2 with $\|f_2\|_\infty < 2^{d+1}p^{1/(d+1)} \approx \|f_1\|_\infty$. The advantage now is that the coefficients of f and g are balanced. The drawback is that the total degree is $n + d + 1$, which is larger than $2d' + 1 \approx d + 1$ for the new method. See Table 8.2 for a comparison for $n = 2$ and $d = 4, 5, 6$.

8.4 Smoothing with two non-linear polynomials

As mentioned in the previous section, when running NFS for finite fields $\mathbb{F}_Q$ with $Q = p^n$ and small $n > 1$, one must use two non-linear polynomials. One might also want to do so for prime fields. Recall that the smoothing step of the individual logarithm stage consists in finding an integer h such that $t^h s$ is $L_Q(2/3, c)$ for a constant $c > 0$. As seen in Section 6.3.5 the descent process introduces exclusively ideals of degree 1 if we start it with a degree 1 ideal. Hence, we require additionally that $t^h s$ splits into degree 1 ideals. When $n = 1$ and g is linear the splitting condition is trivially satisfied as all the ideals in $\mathbb{Q}$ have degree 1. In this section

we discuss the complexity of the smoothing step when f and g are non-linear. We argue that, on the one hand, the naive procedure has a complexity up to $L_Q(2/3, \cdot)$. On the other hand, we show that the adaptation of the rational reconstruction, seen in Section 4.1.2, reduces the complexity of the individual logarithm to its value when no splitting condition is imposed.

Let us recall some notations: $Q = p^n$ is a prime power with $n \geq 1$ to which the NFS algorithm is run; φ is a degree n polynomial of $\mathbb{Z}[x]$ irreducible over $\mathbb{F}_p$ and w a root of φ in $\mathbb{F}_{p^n}$. We call d the degree of the polynomial $f \in \mathbb{Z}[x]$ and we call α a root of f in its number field. We consider the computation of $\log_t s$ when $t = \sum_{i=0}^{n-1} t_i w^i$ and $s = \sum_{i=0}^{n-1} s_i w^i$ in $\mathbb{F}_Q = \mathbb{F}_p(w)$. We denote by identical letters the elements $t = \sum_{i=0}^{n-1} t_i \alpha^i$ and $s = \sum_{i=0}^{n-1} s_i \alpha^i$ in the number field $K = \mathbb{Q}(\alpha)$ of f. For simplicity we can assume here that f is monic.

8.4.1 Inefficiency of the naive approach

Recall that if $P \in \mathbb{Z}[x]$ is a polynomial and ℓ a prime factor of the algebraic number $P(\alpha)$, then the prime ideals of K above ℓ which divide $P(\alpha)$ are of type $\langle \ell, g(\alpha) \rangle$ with g a divisor of $\gcd(\overline{P}(X), \overline{f}(X))$ where the bar denotes the reduction modulo ℓ. In particular, if an algebraic number $a_0 + \alpha a_1 + \cdots + a_{n-1}\alpha^{n-1}$ in K whose norm is divisible by ℓ and $\ell \nmid a_{n-1}$ has a probability of $1/(n-1)!$ that all its ideal factors above ℓ have degree 1. Since smooth norms have a large number of prime divisors, the probability that a random algebraic number has all its prime ideals of degree 1 and of small norm is small. When $n > 1$ this probability depends on the method of polynomial selection that we use and we skip it since we will give an alternative in the following subsection.

Instead let us discuss the case $n = 1$ which is the most unfavorable for the naive approach. Since $n = 1$, the algebraic numbers t and s are in fact integers and, for any h, the element $z := t^h s \mod p\mathcal{O}$ is also an integer, contained in $[0, p-1]$. Hence, it has only degree 1 ideals if and only if its prime divisors are totally split. If z is $L_p(2/3, c)$-smooth, it has in average $\theta\left((\log p)^{1/3}\right)$ prime divisors. Hence, the probability that z is split is less than $P^{\theta\left((\log p)^{1/3}\right)}$ with P the probability that a prime is split in K.

By Frobenius' theorem (see Corollary 3.1.3), P equals $1/\#\operatorname{Gal}(K)$. In the general case we have

$$\#\operatorname{Gal}(K) = \deg(K)! = d^{d(1+o(1))} = \exp(\delta/3(\log p)^{1/3}(\log\log p)^{2/3}).$$

We conclude that the probability that $t^h s \mod p$ has only degree 1 prime ideals is of type $L_p(2/3, \cdot)$; hence a different technique is needed.

8.4.2 Rational reconstruction over number fields

We return to the general case $n \geq 1$. We put $\mathfrak{p} = \langle p, \varphi(\alpha) \rangle$. By rational reconstruction of an element $z \in \mathbb{Z}[\alpha]$ modulo $\mathfrak{p}$ we understand a pair of elements a and b in K such that

$$z \equiv a \cdot b^{-1} \bmod \mathfrak{p}. \tag{8.23}$$

Let us put

$$\mathrm{Lat}(z,\mathfrak{p}) = \left\{ (a_0,\ldots,a_{d-1},b_0,\ldots,b_{d-1}) \in \mathbb{Z}^{2d} \mid \sum_{i=0}^{d-1} a_i\alpha^i \equiv z \left(\sum_{i=0}^{d-1} b_i\alpha^i \right) \bmod \mathfrak{p} \right\}. \tag{8.24}$$

One can check that the rows of the following matrix form a basis of $\mathrm{Lat}(z,\mathfrak{p})$:

$$M = \left(\begin{array}{c|c} \begin{array}{c} p \\ \ddots \\ p \\ \mathrm{vector}(\varphi) \\ \ddots \\ \mathrm{vector}(\varphi) \end{array} & \begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array} \\ \hline \begin{array}{c} \mathrm{vector}(z) \\ \mathrm{vector}(z\alpha) \\ \vdots \\ \mathrm{vector}(z\alpha^{d-1}) \end{array} & \begin{array}{ccc} 1 & & \\ & \ddots & \\ & & 1 \end{array} \end{array} \right), \tag{8.25}$$

where $\mathrm{vector}(z)$ denotes the coordinates of z in basis $(1,\alpha,\ldots,\alpha^{d-1})$.

Note that M has four $d \times d$ blocks and that the number of rows whose unique non-zero entry is p equals n. Hence $\det M = p^n$ and using the LLL algorithm one finds a vector v in $\mathrm{Lat}(z,\mathfrak{p})$ of norm approximatively $Q^{1/2d}$. Hence one can find a rational reconstruction of z modulo $\mathfrak{p}$ with $\mathrm{N}(a)$ and $\mathrm{N}(b) \leq \sqrt{Q}$.

8.4.3 The effect of rational reconstruction

Let $z \equiv ab^{-1} \bmod \mathfrak{p}$ be the rational reconstruction of z modulo $\mathfrak{p}$. Hence we replaced the ideal $z\mathcal{O}$ which has a small probability to have only degree 1 ideals by two random elements a and b of K.

We make the heuristic assumption that a and b are random elements of K of norm less than $\sqrt{Q}$. Let us call L_2 the set of prime ideals of K which have degree 2 or more. One has

$$\mathrm{Prob}\left(\begin{array}{c} \text{an element } \gamma \in K \text{ splits} \\ \text{in degree 1 ideals} \end{array} \right) = \prod_{\mathfrak{l} \in L_2} \left(1 - \frac{1}{\mathrm{N}(\mathfrak{l})} \right). \tag{8.26}$$

But this product is a constant which depends on K, e.g. it equals $6/\pi^2$ when $K = \mathbb{Q}$. Under the heuristic that the smoothness probability and the splitting probability are independent, the complexity of the smoothing step of the individual logarithm stage equals, up to an $1 + o(1)$ exponent, the complexity of the same stage when no splitting condition is imposed.

To sum up, the rational reconstruction allows to

- control the size of the algebraic numbers when $n > 1$;
- replace integers, of bad probability, by algebraic integers, of average probability, when $n = 1$.

Chapter 9

Selecting polynomials for FFS

We continue our journey through discrete logarithms with an improvement to FFS. In Chapter 6 we have mentioned the method of Joux and Lercier [JL02] for selecting appropriate polynomials for FFS. Our goal here is to show that different polynomials can have different performances for the sieve. We discuss the relevance of some criteria which rapidly say if a polynomial is efficient in the FFS relation collection. The Caramel team made recently a series of record computations using FFS in finite fields of characteristic 2. Our criteria have been used to select polynomials for the record computations in [BBD+12, BBD+13]. In this chapter we proceed as follows. We start by making a list of the properties which improve the sieve's yield. We then combine the previously defined functions in order to compare arbitrary polynomials and we show how to rapidly test a large number of candidate polynomials. Next we focus on the case of inseparable polynomials and, in particular, on the Coppersmith algorithm. We conclude with a series of applications of our theoretic results to some examples in the literature.

9.1 Introduction

We presented the function field sieve in Section 6.6.1. The choice of the defining polynomials f and g for the two function fields can be seen as a preliminary stage of the algorithm. It takes a small amount of time but it can greatly influence the sieving stage by slightly changing the probabilities of smoothness. In order to solve the discrete logarithm in $\mathbb{F}_{q^n}$, the main required property of $f, g \in \mathbb{F}_q[t][x]$ is that their resultant $\mathrm{Res}_x(f, g)$ has an irreducible factor $\varphi(t)$ of degree n. Various methods have been proposed to build such polynomials.

The base-m method of polynomial selection, proposed by Adleman [Adl94], consists in choosing $\varphi(t)$ an irreducible polynomial of degree n, setting $g = x - m$, where m is a power of t, and f equal to the base-m expansion of φ. He obtained a sub-exponential complexity of $L_{q^n}(\frac{1}{3}, c)^{1+o(1)}$ with $c = \sqrt[3]{64/9}$. Adleman and Huang [AH99] chose φ to be a sparse polynomial and obtained a constant $c = \sqrt[3]{32/9}$. They also noted that the previously known algorithm of Coppersmith [Cop84] can be seen as a particular case of the FFS. Finally, Joux and Lercier [JL02] introduced a method which, without improving the complexity, behaves

better in practice. Presented in Algorithm 6.6, it consists in selecting a polynomial f with small degree coefficients and then of randomly testing linear polynomials $g = g_1 x + g_0$ until $\mathrm{Res}_x(f, g)$ has an irreducible factor of degree n. In [JL06] Joux and Lercier proposed two additional variants of their methods. In the first one, the *Two rational sides* that we presented in Section 5.4, we add the condition that f has degree 1 in t. Its main advantage is that its description does not require the theory of function fields. The second variant, called the *Galois improvement*, applies to the case where n is composite.

In this chapter we improve the method of Joux and Lercier [JL02] by showing how to select the non-linear polynomial f. For that we follow the strategy that was developed in the factorization context. In particular Murphy [Mur99] introduced and used criteria which allow to rapidly rank any set of polynomials. See [Bai11], for recent developments in this direction.

Therefore, we introduce relevant functions for the sieving efficiency of a polynomial, taking into account a size property, a so-called root property that reflects the behaviour modulo small irreducible polynomials, and a cancellation property, which is analogous to the real roots property for the NFS. We also present efficient algorithms for quantifying these properties. A special attention is given to the particular case where f is not separable. Indeed, this is a phenomenon that has no analogue in the factorization world and that has strong repercussions on the sieving efficiency.

9.2 Quantification functions

9.2.1 Size property

We start by deciding the degrees in t and x of the two polynomials f and g. The FFS has always been implemented for polynomials f of small coefficients in t and for polynomials g of degree 1 in x, like in [JL02]. It might not be obvious that this is the best choice. For instance in the case of the NFS both for factorization [Mon06, PZ11] and discrete logarithm [JL03], pairs of non-linear polynomials were used. In the following, we argue that the classical choice is indeed the best one.

Let us first recall the nature of the objects we have to test for smoothness. In FFS one collects coprime pairs $(a(t), b(t)) \in \mathbb{F}_q[t]$ such that the norms of $a - bx$ in the function fields of f and g are both smooth. These norms are polynomials in t of a simple form: denoting $F(X, Y) = f(\frac{X}{Y})Y^{\deg_x f}$ the homogenization of $f(x)$, the norm of $a - bx$ is just $F(a, b)$. Similarly, we denote $G(X, Y)$ the homogenization of $g(x)$ and the second norm is $G(a, b)$. As a consequence, the polynomial selection stage can be restated as the search for polynomials f and g such that $F(a, b)$ and $G(a, b)$ are likely to be both smooth for coprime pairs (a, b) in a given range.

As a first approximation, we translate this condition into the fact that the degree of the product $F(a, b)G(a, b)$ is as small as possible. It will be refined all along this chapter.

Fact 9.2.1. *Assume that we have to compute discrete logarithms in $\mathbb{F}_{q^n}$, with polynomials $f, g \in \mathbb{F}_q[t][x]$, such that $\deg_x g \leq \deg_x f$. If, for a and b in a set*

of polynomials given by an upper bound e on the degree, the polynomials f and g minimize $\max\{\deg F(a,b)G(a,b)\}$*, then we have* $\deg_x g = 1$.

Argument. Let (a,b) be any pair of degree e, and assume that there is no cancellation of the monomials in $F(a,b)$ and $G(a,b)$ respectively. Then one has

$$\deg\big(F(a,b)G(a,b)\big) = \deg_t g + e\deg_x g + \deg_t f + e\deg_x f. \tag{9.1}$$

The degree of the resultant of f and g can be bounded by $\deg \mathrm{Res}(f,g) \leq \deg_x f \deg_t g + \deg_x g \deg_t f$. Since we need this resultant to be of degree at least n, we impose

$$\deg_x f \deg_t g + \deg_x g \deg_t f \geq n. \tag{9.2}$$

For a fixed value of $\deg_x f$ and $\deg_x g$ we vary $\deg_t f$ and $\deg_t g$ without changing the left hand side in Equation 9.2. Since $\deg_x g \leq \deg_x f$, we minimize the expression in Equation 9.1 when we minimize $\deg_t f$. Since this is independent on $\deg_x f$ and $\deg_x g$, in particular in the optimal configuration of the degrees of f and g we have $\deg_t f \approx 0$. Therefore we set $\deg_t f$ as small as possible, let us call this degree ε. Hence the optimization problem becomes

$$\begin{aligned} \text{minimize} \quad & (\deg_x f + \varepsilon + e\deg_x g + \deg_t g) \\ \text{when} \quad & \deg_x f \deg_t g + \varepsilon \deg_x g \geq n. \end{aligned}$$

Since one can decrease $\deg_x g$ without changing too much the left hand side of the constraint, the choice $\deg_x g = 1$ is optimal. □

Of course the numerous simplifications can make that, for some values of n, there exist pairs of polynomials which are slightly better without satisfying $\deg_x g = 1$. Nevertheless, for a generic values of n, we expect that the best pairs (f,g) have a linear g. In the rest of the article we simply write d for $\deg_x f$. The degree of g in t is then about n/d.

Remark 9.2.2. We decided to optimize the degree of the product of the norms. In terms of smoothness probability, it is only pertinent if both norms have similar degrees. More precisely, as the logarithm of Dickman's rho function is concave (see Figure 4.1 and Section 1.1), it can be shown that it is optimal to balance the degrees of the norms. Hence sensible choices of the parameters are such that $de \approx \frac{n}{d} + e$.

We are now ready to quantify the size property for a single polynomial f. It clearly depends on the bound e on both $\deg a$ and $\deg b$. In the following definition, we also take into account the skewness improvement as implemented in [DGV13], that is, we set the skewness s to be the difference between the bounds on the degree of a and of b. We can then define sigma to match the average degree of $F(a,b)$ for (a,b) in a domain of polynomials of degrees bounded by $\lfloor e + s/2 \rfloor$ and $\lfloor e - s/2 \rfloor$, when no cancellation occurs among the monomials of $F(a,b)$. This translates into the following formal definition.

Definition 9.2.3. Let $f \in \mathbb{F}_q[t][x]$ be a polynomial, s the skewness parameter and e the sieve size parameter. We define the size property of f as

$$\sigma(f,s,e) = \sum_{\substack{0 \le d_a \le e+s/2 \\ 0 \le d_b \le e-s/2}} p_{d_a,d_b} \max_{i\in[0,d]} \left(\deg(f_i) + id_a + (d-i)d_b \right), \quad (9.3)$$

with $p_{d_a,d_b} = (q-1)^2 q^{d_a+d_b}/q^{\lfloor e+s/2 \rfloor + \lfloor e-s/2 \rfloor + 2}$.

9.2.2 Root property

As a first approximation, for a random pair $(a,b) \in \mathbb{F}_q[t]^2$, $F(a,b)$ has a smoothness probability of the same order of magnitude as random polynomials of the same degree. Nevertheless, we will show that for a fixed size property, some polynomials improve this probability by a factor of 2 or more.

Consider the example of $f = x(x-1) - (t^{64} - t)$ in $\mathbb{F}_2[t][x]$. For all monic irreducible polynomials ℓ of degree at most 3, ℓ divides the constant term, so f has two roots modulo ℓ. For each such ℓ and for all b not divisible by ℓ, there are 2 residues of a modulo ℓ such that $F(a,b) \equiv 0 \mod \ell$. Therefore, for all ℓ of degree 3 or less, the probability that ℓ divides the norm is heuristically twice as large as the probability that it divides a random polynomial. This influences in turn the smoothness probability. This effect is quantified by the function alpha that we now introduce.

Definition of alpha

Introduced by Murphy [Mur99] in the case of the NFS, alpha can be extended to the case of the FFS. Let ℓ be a monic irreducible polynomial in $\mathbb{F}_q[t]$ and let us call ℓ-part of a polynomial P, the largest power of ℓ in P. We will prove that the quantity α_ℓ below is the degree difference between the ℓ-part of a random polynomial and the ℓ-part of $F(a,b)$ for a random coprime pair $(a,b) \in \mathbb{F}_q[t]$.

Let us first properly define the average of a function on a set of polynomials.

Definition 9.2.4. Let v be a real function of one or two polynomial variables $v : \mathbb{F}_q[t] \to \mathbb{R}$ or $v : \mathbb{F}_q[t] \times \mathbb{F}_q[t] \to \mathbb{R}$. Let S be a subset of the domain of v. For any pair (a,b) of polynomials, we write $\deg(a,b)$ for $\max(\deg(a), \deg(b))$. If the limit below exists we call it average of v over S and denote it by $\mathbb{A}(v,S)$:

$$\mathbb{A}(v,S) = \lim_{N\to\infty} \frac{\sum_{s\in S, \deg(s)\le N} \varphi(s)}{\#\{s \in S \mid \deg(s) \le N\}}.$$

Definition 9.2.5. Let L denote the set of monic irreducible polynomials in $\mathbb{F}_q[t]$. Put $D = \{(a,b) \in \mathbb{F}_q[t]^2 \mid \gcd(a,b) = 1\}$. Take a non-constant polynomial $f \in \mathbb{F}_q[t][x]$. When the right hand members are defined, we set for all $\ell \in L$:

$$\begin{aligned} \alpha_\ell(f) &= \deg(\ell) \left(\mathbb{A}\left(v_\ell(P), \{P \in \mathbb{F}_q[t]\}\right) - \mathbb{A}\left(v_\ell(F(a,b)), \{(a,b) \in D\}\right) \right), \\ \alpha(f) &= \sum_{\ell \in L} \alpha_\ell(f), \end{aligned}$$

where v_ℓ is the valuation at ℓ. The infinite sum which defines $\alpha(f)$ must be seen as a formal notation and by its sum we denote the limit when b_0 goes to infinity of $\alpha(f, b_0) := \sum_{\ell \in L, \deg \ell \leq b_0} \alpha_\ell(f)$.

Notation 9.2.6. For all irreducible polynomial $\ell \in \mathbb{F}_q[t]$, $\mathrm{N}(\ell)$ denotes the number of residues modulo ℓ, i.e., $q^{\deg \ell}$.

We call affine root of f modulo ℓ^k any $r \in \mathbb{F}_q[t]$ such that $\deg r < k \deg \ell$ and $F(r, 1) = f(r) \equiv 0 \mod \ell^k$. Also we call projective root of f modulo ℓ^k the polynomials $r \in \mathbb{F}_q[t]$ such that $\ell \mid r$, $\deg r < k \deg \ell$ and $F(1, r) \equiv 0 \mod \ell^k$. Note that, when $k = 1$ any affine and projective roots can be seen as an element of $\mathbb{P}^1(\mathbb{F}_q(t))$, hence we denote them $(r : 1)$ and $(1 : r)$ respectively.

Notation 9.2.7. We denote by $S(f, \ell)$ the set of affine and projective roots modulo ℓ^k for any k:

$$S(f, \ell) = \left\{ (r, k) \mid k \geq 1, r \text{ affine or projective root of } f \text{ modulo } \ell^k \right\}. \tag{9.4}$$

Proposition 9.2.8. *Let $f \in \mathbb{F}_q[t][x]$ and $\ell \in \mathbb{F}_q[t]$ a monic irreducible polynomial. Then α_ℓ exists and we have*

$$\alpha_\ell(f) = \deg \ell \left(\frac{1}{\mathrm{N}(\ell) - 1} - \frac{\mathrm{N}(\ell)}{\mathrm{N}(\ell) + 1} \sum_{(r,k) \in S(f,\ell)} \frac{1}{\mathrm{N}(\ell)^k} \right). \tag{9.5}$$

Proof. In order to prove the convergence of Equation 9.5, note that some elements of $S(f, \ell)$ group into infinite sequences $\{(r^{(k)}, k)\}_k$ with $r^{(k)} \equiv r^{(k-1)} \mod \ell^{k-1}$. Since each infinite sequence defines a root of f in the ℓ-adic completion of $\mathbb{F}_q(t)$, there are at most d such sequences, whose contributions converge geometrically. There are only finitely many remaining elements of $S(f, \ell)$ because otherwise one could extract an additional ℓ-adic root. This proves the convergence.

In order to show the equality, note that we have $\mathbb{A}(v_\ell(P), \mathbb{F}_q[t]) = \sum_{k=1}^{\infty} \frac{1}{\mathrm{N}(\ell)^k} = \frac{1}{\mathrm{N}(\ell)-1}$. Indeed, for all $k \in \mathbb{N}$, the density of the set of polynomials divisible by ℓ^k is the inverse of the number of residues modulo ℓ^k, which is $q^{\deg(\ell)k} = \mathrm{N}(\ell)^k$.

Let us compute $a_h^{(\ell)} := \mathbb{A}(v_\ell(F(a, b)), \{(a, b) \in \mathbb{F}_q[t]^2 \mid \gcd(a, b) \not\equiv 0 \bmod \ell\})$. This corresponds to the contribution of $F(a, b)$ in the definition of α_ℓ. The condition $\gcd(a, b) = 1$ has been replaced by a local condition. Since we are only interested in ℓ-valuations, this does not change the result. The number of ℓ-coprime pairs (a, b) of degree less than $k \deg \ell$ is $\mathrm{N}(\ell)^{2k} - \mathrm{N}(\ell)^{2k-2}$. Such a pair (a, b) satisfies $\ell^k \mid F(a, b)$ if and only if $\ell \nmid b$ and ab^{-1} is an affine root modulo ℓ^k or $\ell \mid b$ and ba^{-1} is a projective root modulo ℓ^k. For each affine root r, the number of ℓ-coprime pairs (a, b) such that $a \equiv br \mod \ell^k$ equals the number of choices for b, which is $\mathrm{N}(\ell)^k - \mathrm{N}(\ell)^{k-1}$. Similarly, for each projective root, the number of coprime pairs (a, b) such that $b \equiv ar \mod \ell^k$ is the number of choices for a, which is $\mathrm{N}(\ell)^k - \mathrm{N}(\ell)^{k-1}$. Hence, it follows that for each $(r, k) \in S(f, \ell)$, the probability that the residues of a coprime pair (a, b) modulo ℓ^k match the root (r, k) is given by the formula below, where $(a : b) \equiv r \mod \ell^k$ is short for $r \equiv ab^{-1} \mod \ell^k$ when $\ell \nmid b$ or $r \equiv ba^{-1} \mod \ell^k$ when $\ell \mid b$.

$$\mathrm{Prob}((a : b) \equiv r \bmod \ell^k) = \frac{1}{\mathrm{N}(\ell)^k} \frac{\mathrm{N}(\ell)}{\mathrm{N}(\ell) + 1}, \tag{9.6}$$

Let us complete the proof using the following equation, that we prove in Lemma 9.2.10:

$$\begin{array}{rcl} a_h^{(\ell)}(f) & = & \sum_{k=1}^{\infty} k \operatorname{Prob}(v_\ell(F(a,b)) = k) \\ = \sum_{k=1}^{\infty} \operatorname{Prob}(v_\ell(F(a,b)) \geq k) & = & \sum_{(r,k)\in S(f,\ell)} \operatorname{Prob}((a:b) \equiv r \bmod \ell^k). \end{array} \tag{9.7}$$

When replacing $a_h^{(\ell)}$ in formula $\alpha_\ell(f) = \frac{\deg \ell}{\mathrm{N}(\ell)-1} - \deg(\ell) a_h^{(\ell)}(f)$ and using Equation 9.6 we obatin the desired result.

□

If f has only simple roots modulo ℓ, then by Hensel's Lemma f has the same number of roots modulo every power of ℓ. We obtain the following formula.

Corollary 9.2.9. *Let $f \in \mathbb{F}_q[x][t]$ and ℓ a monic irreducible polynomial in $\mathbb{F}_q[t]$ such that the affine and projective roots of f modulo ℓ are simple and call n_ℓ their number. Then*

$$\alpha_\ell(f) = \frac{\deg \ell}{\mathrm{N}(\ell) - 1} \left(1 - \frac{\mathrm{N}(\ell)}{\mathrm{N}(\ell)+1} n_\ell\right). \tag{9.8}$$

Let us come back to Equationn (9.7). The technicalities arise from the fact that the "probability" sign stands for the natural density, which is not a probability.

Lemma 9.2.10. *Under the notations of Proposition 9.2.8, Equation 9.7 holds.*

Proof. We prove the equality of the four members of Equation (9.7) from right to left. First, the equality below holds as we can change the summation order of an absolutely convergent series.

$$\sum_{k=1}^{\infty} \operatorname{Prob}\left(v_\ell(F(a,b)) \geq k\right) = \sum_{(r,k)\in S(f,\ell)} \operatorname{Prob}\left((a:b) \equiv r \bmod \ell^k\right).$$

Secondly, for large enough k the roots modulo ℓ^k are simple, so $k \operatorname{Prob}(\ell^k \mid F(a,b)) \to 0$ when k goes to infinity. Since for all k, $\operatorname{Prob}(v_\ell(F(a,b)) = k) = \operatorname{Prob}(\ell^k \mid F(a,b)) - \operatorname{Prob}(\ell^{k+1} \mid F(a,b))$, we have

$$\sum_{k=1}^{\infty} k \operatorname{Prob}\left(v_\ell(F(a,b)) = k\right) = \sum_{k=1}^{\infty} \operatorname{Prob}\left(v_\ell(F(a,b)) \geq k\right).$$

Finally, let us prove that $a_{\hom}(f)$ exists and equals $\sum_{k=1}^{\infty} k \operatorname{Prob}\left(v_\ell(F(a,b)) = k\right)$. For each N and k put

$$a_{\hom}(f; N, k) = \frac{\sum\{\min(v_\ell(F(a,b)), k) \mid \gcd(a,b) \not\equiv 0 \bmod \ell, \deg a, \deg b \leq N\}}{\#\{(a,b) \mid \gcd(a,b) \not\equiv 0 \bmod \ell, \deg a, \deg b \leq N\}}.$$

Call $a_{\hom}(f; N)$ the expression above when $\min(v_\ell(F(a,b)), k)$ is replaced with $v_\ell(F(a,b))$. On the one hand, for any $k_0 \in \mathbb{N}$, we have $a_{\hom}(f; N, k_0) \leq a_{\hom}(f; N)$, so

$$\sum_{k=1}^{k_0} k \operatorname{Prob}(v_\ell(F(a,b)) = k) \leq \liminf_{N\to\infty} a_{\hom}(f; N).$$

On the other hand, let k_1 be large enough such that all the affine and projective roots modulo ℓ^{k_1} are simple and put $N = k_1 \deg \ell$. For this value of N, the proportion of pairs (a, b) of degree at most N such that $(a : b) \equiv r \mod \ell^k$ for a root (r, k) equals the probability $\text{Prob}((a : b) \equiv r \mod \ell^k)$. Hence $a_{\text{hom}}(f; N, N/\deg \ell) = \sum_{k=1}^{k_1} k \,\text{Prob}(v_\ell(F(a, b)) = k)$. Now, since the roots modulo ℓ^{k_1} are simple there are at most $\deg f$ of them. Also, for a and b of degree bounded by N, the norm $F(a, b)$ has degree bounded by $N \deg_x f + \deg_t f$. Hence we obtain

$$|a_{\text{hom}}(f; N) - a_{\text{hom}}(f; N, N/\deg \ell)| \leq \frac{\deg_x f(N \deg_x f + \deg_t f)}{q^{N/\deg \ell}}.$$

This further implies

$$\limsup_{N\to\infty} a_{\text{hom}}(f; N) \leq \sum_{k\in\mathbb{N}} k \,\text{Prob}(v_\ell(F(a, b)) = k).$$

We conclude that $a_{\text{hom}}(f; N)$ converges to $\sum_{k\in\mathbb{N}} k \,\text{Prob}(v_\ell(F(a, b)) = k)$. □

The case of linear polynomials

Showing that $\alpha(f)$ converges is not trivial and, to our knowledge, it is not proven in the NFS case. A joint work with Armand Lanchand, which is in preparation, studies $\alpha(f)$ in the NFS case. Let us first show that $\alpha(g)$ converges for linear polynomials g. This requires the following classical identity.

Lemma 9.2.11. *(Chapter I,[Apo90]) Let μ denote Möbius' function and let x be such that $|x| < 1$. Then we have*

$$\sum_{h\geq 1} \mu(h) \frac{x^h}{1 - x^h} = x.$$

Notation 9.2.12. We denote by I_k the number of monic degree-k irreducible polynomials in $\mathbb{F}_q[t]$.

Theorem 9.2.13. *Let $g \in \mathbb{F}_q[t][x]$ be such that $\deg_x g = 1$. Then*

$$\alpha(g) = \frac{1}{q - 1}.$$

Proof. Let L be the set of monic irreducible polynomials in $\mathbb{F}_q[t]$. We will prove that, when b_0 goes to infinity, $\sum_{\ell\in L, \deg \ell\leq b_0} \alpha_\ell(g)$ tends to $\frac{1}{q-1}$. In Equation 9.8 one has $n_\ell = 1$ and therefore

$$\sum_{\ell\in L, \deg \ell\leq b_0} \alpha_\ell(g) = \sum_{\ell\in L, \deg \ell\leq b_0} \frac{\deg(\ell)}{\text{N}(\ell)^2 - 1} = \sum_{k\leq b_0} \frac{kI_k}{q^{2k} - 1}. \tag{9.9}$$

Since $kI_k = \sum_{h|k} \mu(h) q^{\frac{k}{h}}$, the series transforms into a double series for which we will prove the absolute convergence and will compute the sum:

$$\sum_{k\geq 1} \sum_{h|k} \frac{1}{q^{2k} - 1} \mu(h) q^{\frac{k}{h}}.$$

The absolute value of the term $\left|\mu(h)\frac{q^{\frac{k}{h}}}{q^{2k}-1}\right|$ is bounded by $\frac{q^{\frac{k}{h}}}{q^{2k}-1}$. It follows easily that the sum is bounded by $\frac{4}{q-1}$. Therefore, we can change the summation order:

$$\sum_{k\geq 1}\sum_{h|k}\mu(h)\frac{q^{\frac{k}{h}}}{q^{2k}-1} = \sum_{i\geq 1}\left(\sum_{h\geq 1}\mu(h)\frac{q^{i}}{q^{2hi}-1}\right). \tag{9.10}$$

When applying Lemma 9.2.11 to $x = \frac{1}{q^{2i}}$ we obtain $\sum_{h\geq 1}\mu(h)\frac{1}{q^{2hi}-1} = \frac{1}{q^{2i}}$. This shows that, when b_0 goes to infinity, $\sum_{\deg \ell\leq b_0}\alpha_\ell(g)$ tends to $\sum_{i\geq 1}\frac{1}{q^i} = \frac{1}{q-1}$. □

We conclude the subsection by showing the convergence of alpha. Let now $\mathcal{C}$ be a (singular or non-singular) projective curve. Call $\mathcal{P}_k(\mathcal{C})$ the set of points of $\mathcal{C}$ with coefficients in $\mathbb{F}_{q^k}$ and $P_k(\mathcal{C})$ its cardinality. Next, call $\mathcal{P}'_k(\mathcal{C})$ the set of points in $\mathcal{P}_k(\mathcal{C})$ whose t-coordinate does not belong to a strict subfield of $\mathbb{F}_{q^k}$. Finally, we denote by $P'_k(\mathcal{C})$ its cardinality.

We will need the following intermediate result.

Lemma 9.2.14. *Let $\mathcal{C}$ be projective plane curve of degree d_0 defined over $\mathbb{F}_q$ and let $g_0 = (d_0-1)(d_0-2)/2$ be its arithmetic genus. Then, for all $k \geq 1$,*

$$\left|P'_k(\mathcal{C}) - (q^k+1)\right| < (4g_0+6)q^{\frac{k}{2}}. \tag{9.11}$$

Proof. Let $\tilde{\mathcal{C}}$ be a non-singular model for $\mathcal{C}$ and let g be its geometric genus. We apply the Hasse-Weil Theorem and obtain:

$$\left|P_k(\tilde{\mathcal{C}}) - (q^k+1)\right| \leq 2g\cdot q^{\frac{k}{2}}. \tag{9.12}$$

Next, according to Chapter VI in [Ful69],

$$|P_k(\tilde{\mathcal{C}}) - P_k(\mathcal{C})| \leq g_0 - g. \tag{9.13}$$

Every point (t_0, x_0) of $\mathcal{P}_k(\mathcal{C})\backslash\mathcal{P}'_k(\mathcal{C})$ is determined by the choice of: a) a strict divisor d of k, b) an element $t_0 \in \mathbb{F}_{q^{\frac{k}{d}}}$ and c) a root x_0 of $f(t_0, x)$ in $\mathbb{F}_{q^k}$. Therefore:

$$\left|P'_k(\mathcal{C}) - P_k(\mathcal{C})\right| \leq \sum_{d|k,1<d\leq k} q^{\frac{k}{d}}\deg_x(f). \tag{9.14}$$

On the one hand $\deg_x f < g_0 + 3$; on the other hand, if one calls d_1 the smallest proper divisor of k, $\sum_{d|k,1<d\leq k} q^{\frac{k}{d}} = q^{\frac{k}{d_1}}\sum_{d|k,1<d\leq k} q^{\frac{k}{d}-\frac{k}{d_1}}$. Since $d \mid k$ and $d \geq d_1$, $\frac{k}{d} - \frac{k}{d_1}$ is a negative or zero integer. Therefore this sum is bounded by $q^{\frac{k}{d_1}}\sum_{i\geq 0} q^{-i} \leq q^{\frac{k}{d_1}}\sum_{i\geq 0} 2^{-i} = 2q^{\frac{k}{d_1}}$. But $d_1 \geq 2$, so

$$\left|P'_k(\mathcal{C}) - P_k(\mathcal{C})\right| \leq 2q^{\frac{k}{2}}\deg_x f < 2(g_0+3)q^{\frac{k}{2}}. \tag{9.15}$$

The result follows from Equations 9.12, 9.13 and 9.15. □

The following theorem shows that $\alpha(f)$ is well defined. More specifically, call L the set of irreducible monic polynomials in $\mathbb{F}_q[t]$. We show the existence of alpha by showing the convergence of $v_{b_0} := \sum_{\ell \in L, \deg \ell \leq b_0} (\alpha_\ell(f) - \alpha_\ell(x))$.

Theorem 9.2.15. *Let $f \in \mathbb{F}_q[t][x]$ an absolutely irreducible separable polynomial. Then the sequence v_{b_0} defined above converges. If one defines alpha by $\alpha(f) = \lim_{b_0 \to \infty} \sum_{\deg \ell \leq b_0, \ell \in L} \alpha_\ell(f)$, then there exist explicit bounds on α that depend only on $\deg_x f$, $\deg_t f$ and q.*

Proof. Call $d_0 = \deg f$ the degree of f as a polynomial in two variables and $g_0 = (d_0 - 1)(d_0 - 2)/2$. Let L_0 be the set of irreducible divisors of $\mathrm{Disc}(f) \cdot f_d$. Call b_0 the largest degree of elements in L_0. Let $k > b_0$. We are in the case of Corollary 9.2.9, hence Equation 9.8 gives

$$\sum_{\ell \in L, \deg \ell = k} \alpha_\ell(f) - \alpha_\ell(x) =$$
$$= \frac{kq^k}{q^{2k} - 1} \left(\#\{(\ell, r) \mid \deg(\ell) = k, f(r) \equiv 0 \mod \ell\} - I_k \right).$$

Each pair (ℓ, r) as in the equation above corresponds to exactly k points on the curve $\mathcal{C}$ associated to f. Indeed, each ℓ has exactly k distinct roots in $\mathbb{F}_{q^k}$. Hence we have $I_k = \frac{1}{k} P'_k(\mathbb{P}^1(\mathbb{F}_q))$ and $\#\{(\ell, r) \mid f(r) \equiv 0 \mod \ell\} = \frac{1}{k} P'_k(\mathcal{C})$, and further:

$$\left| \#\{(\ell, r) \mid f(r) \equiv 0 \mod \ell\} - I_k \right| = \frac{1}{k} \left| P'_k(\mathcal{C}) - P'_k(\mathbb{P}^1(\mathbb{F}_q)) \right|. \tag{9.16}$$

Finally, Lemma 9.2.14 applied to $\mathcal{C}$ and $\mathbb{P}^1(\mathbb{F}_q))$ respectively gives

$$\left| P'_k(\mathcal{C}) - (q^k + 1) \right| \leq (4g_0 + 6)\sqrt{q^k} \tag{9.17}$$
$$\left| P'_k(\mathbb{P}^1(\mathbb{F}_q)) - (q^k + 1) \right| \leq 6\sqrt{q^k}, \tag{9.18}$$

where g_0 is the arithmetic genus of $\mathcal{C}$.

Hence $\left| \#\{(\ell, r) \mid f(r) \equiv 0 \mod \ell\} - I_k \right| \frac{kq^k}{q^{2k}-1} \leq (4g_0 + 12) \frac{q^k}{q^{2k}-1} \sqrt{q^k}$. The series $\sum_{k \geq 1} \frac{q^k}{q^{2k}-1} \sqrt{q^k}$ is equivalent to the series $\sum_k \sqrt{q}^{-k}$ which converges. Therefore the sequence v_{b_0} converges when b_0 tends to infinity.

For a given pair d_0 and q, one can clearly bound the set L_0. For all $\ell \in L_0$, by Proposition 9.2.8, $S(f, \ell)$ is formed by at most $\deg_x f \leq d_0$ infinite sequences and a finite number of additional elements. We are thus left with finding a bound for the roots which do not extend into ℓ-adic roots. By Hensel's Lemma, if a root (r, k) does not lift to an ℓ-adic one, we have $f'(r) \equiv 0 \mod \ell^k$. This implies $\mathrm{Disc}(f) \equiv 0 \mod \ell^k$ which gives a bound on k. Therefore, alpha admits an effective bound depending exclusively on q and d_0. □

Example 9.2.16. Following the proof of the previous theorem we can, in addition to finding a bound on α, evaluate the speed of convergence. Take $q = 2$, and let $f \in \mathbb{F}_q[t][x]$ such that $\deg_x f = 6$ and $\tilde{g} = 19$ and suppose that L_0 contains only polynomials of degree less than 15. Using Equations 9.16 and 9.17 in the proof above and the exact formula for I_k we can prove that $\alpha(f)$ is computed up to an error of 0.567 if we sum polynomials ℓ up to degree 15 and we reduce the error to 0.097 if we go to degree 20.

9.2.3 Cancellation property-Laurent roots

Consider the polynomial $f = x^3 + t^2x + 1 \in \mathbb{F}_2[t][x]$. For all $(a, b) \in \mathbb{F}_2[t]^2$, if no cancellations occur, the degree of $F(a, b) = a^3 + t^2ab^2 + b^3$ is $\max(\deg a^3, \deg(t^2ab^2), \deg b^3)$. One can easily check that the degree of $F(a, b)$ is lower than this value if and only if $\deg a - \deg b$ equals 1 or -2. Moreover, in the first case the decrease is at least 2 while in the second case it is at least 1. These conditions can be better explained thanks to the Laurent series.

We call Laurent series (in $1/t$) over $\mathbb{F}_q$ any series $\sum_{n\geq n_0} a_n \frac{1}{t^n}$ with $n_0 \in \mathbb{Z}$ and coefficients a_n in $\mathbb{F}_q$. We make the common convention to call degree of a rational fraction f_1/f_2 with $f_1, f_2 \in \mathbb{F}_q[t]$ the difference $\deg f_1 - \deg f_2$. The degree of a Laurent series is then defined as the degree of any of its nonzero truncations e.g $t + 1 + 1/t^2 + \cdots$ has degree 1. Equivalently, it is the opposite of the valuation of the Laurent series in $1/t$. We call Laurent polynomial a pair (r, m) such that $r \in \mathbb{F}_q(t)$, m is an integer and r is the sum of a Laurent series whose terms are zero starting from index $m + 1$. We may also use $r + O(\frac{1}{t^{m+1}})$ for writing the Laurent polynomial (r, m).

Formally, a pair (a, b) has a "decrease in degree" if $\max_i \deg(f_i a^i b^{d-i})$ is strictly larger than $\deg F(a, b)$. Note that $F(a, b)$ has a decrease in degree if and only if $b \neq 0$ and the first terms of the Laurent series $\frac{a}{b}$ match those of a Laurent polynomial r with the property given in the following definition.

Definition 9.2.17. Let $f \in \mathbb{F}_q[t][x]$ be a polynomial and call d its degree in x. Let (r, m) be a Laurent polynomial. We say that (r, m) is a Laurent root of f if

$$\max_{i\in[0,d]} \deg(f_i r^i) - \deg f(r) > 0. \tag{9.19}$$

We call gap of (r, m) the least value in the left hand side of the inequality above when we replace r by any Laurent series extending r. A Laurent series such that all its truncations are Laurent roots is called an infinite Laurent root.

In the example above, $\frac{1}{t^2} + \frac{1}{t^8} + O(\frac{1}{t^9})$ is a Laurent root of gap 7 and it extends into an infinite Laurent root. Also $t + O(1)$ is a Laurent root of gap 2 that is not the truncation of any Laurent root (r, m) with $m \geq 2$ (for $m = 1$ it extends in $t + O(\frac{1}{t})$ which has the same gap). Note that the gap is not directly connected to the number of terms in the Laurent polynomial.

Computation

One can compute every Laurent root in two steps. First, one computes the Laurent roots of type λt^δ with λ is in $\mathbb{F}_q$ and δ is an integer. For this, call Newton polygon of f, with respect to valuation $-\deg$, the convex hull of $\{(d - i, \deg(f_i)) \mid i \in [0, d]\}$. Chapter II in [Neu99] shows that δ must be an integer slope of the Newton polygon of f. Next, to extend a Laurent root $r = a_{n_0}t^{n_0} + \cdots + a_m\frac{1}{t^m}$ with $a_m \neq 0$ to a root with precision larger than m, one computes the Laurent roots λt^δ of $f(x + r)$ for which δ is an integer such that $\delta < -m$. Note that this corresponds to a Hensel lift with respect to the valuation $-\deg$.

In order to compute the gap of a Laurent root (r, m), note that in Equation 9.19 the term $\max_{i\in[0,d]} \deg(f_i r^i)$ depends only on the leading term of r. Hence the

problem is reduced to that of computing the maximal degree of $f(R)$ for the Laurent series R which extend r. For this, one sets an upper bound and then tests Laurent polynomials with increasingly more terms and reduces the upper bound until they produce a certificate.

Definition of α_∞.

For each Laurent root (r, m), we can compute the proportion of pairs (a, b) on a sieve domain such that the first terms of a/b match (r, m). Recall that a sieve domain of sieve parameter e and skewness s corresponds to all the pairs (a, b) with $\deg a \leq \lfloor e + s/2 \rfloor$ and $\deg b \leq \lfloor e - s/2 \rfloor$.

Lemma 9.2.18. *Let $r + O(\frac{1}{t^{m+1}})$ be a Laurent root of $f \in \mathbb{F}_q[t][x]$. Call $N_r = \deg(r) + m$, the number of terms of r other than the leading one. Then the proportion of pairs (a, b) on a domain of sieve parameter $e \geq N_r + |\deg r|$ and skewness parameter s such that the Laurent series a/b matches $r + O(\frac{1}{t^{m+1}})$ is:*

$$\text{Prob}\left(\frac{a}{b} = r + O\left(\frac{1}{t^{m+1}}\right)\right) = \frac{q^{-N_r - |s - \deg r|}}{q+1}\left(1 + O_{e\to\infty}(1/q^{2e})\right).$$

Proof. The proportion of pairs (a, b) such that $\deg a - \deg b = \deg r$ is approximated by $(\frac{q-1}{q})^2 \cdot \sum_{i\geq 0} \frac{1}{q^{2i}} q^{-|s-\deg r|} = \frac{q-1}{q+1} q^{-|s-\deg r|}$. See Figure 9.1 for an illustration. The relative error made is $O(1/q^{2e})$ and corresponds to the fact that the series $\sum \frac{1}{q^{2i}}$ must be truncated at $i = e - |\deg r|$ and to the fact that for $\deg a, \deg b < |s|$ there is no pair (a, b) such that $\deg a - \deg b = s$. Next, only a fraction $\frac{1}{q-1}$ of these pairs have leading coefficients such that $a/b = r(1 + O(\frac{1}{t}))$. Finally, when a/b matches the leading term of r, the condition that it matches the other terms of r, $a/b = r + O(\frac{1}{t^{m+1}}) = r\left(1 + O(\frac{1}{t^{N_r+1}})\right)$, can be expressed as a system of N_r linear equations which is triangular on the variables of a. Therefore, a pair with $\deg a - \deg b = \deg r$ and $a/b = r(1 + O(1/t))$ has probability q^{-N_r} to satisfy $a/b = r(1 + O(\frac{1}{t^{N_r+1}}))$. This leads to the proportion announced in the statement. Note that the condition $e > N_r + |\deg r|$ guarantees that the polynomials a and b have sufficiently many coefficients so that the linear conditions make sense. □

We can now define a function which, for large sieve domains, measures the average degree gained due to the cancellations.

Definition 9.2.19. For any Laurent root $r + O(\frac{1}{t^{m+1}})$ we call $\text{trunc}(r)$ the Laurent polynomial obtained by deleting the term in $\frac{1}{t^m}$. If $\text{trunc}(r) \neq 0$ we write $\gamma(r, m)$ for the gap of $r + O(\frac{1}{t^{m+1}})$ minus the gap of $\text{trunc}(r) + O(\frac{1}{t^m})$. Otherwise $\gamma(r, m)$ is the gap of $r + O(\frac{1}{t^{m+1}})$. We call alpha infinity the following quantity

$$\alpha_\infty(f, s) := \sum_{(r,m) \text{ Laurent root}} -\gamma(r, m) \frac{1}{q+1} q^{-N_r - |s - \deg(r)|}. \tag{9.20}$$

Consider the case when all the Laurent roots of f extend infinitely into the Laurent series r_1, r_2, ..., r_h. If, for all i, each new term of a/b which matches r_i increases the gap by one, then we obtain the simpler formula below.

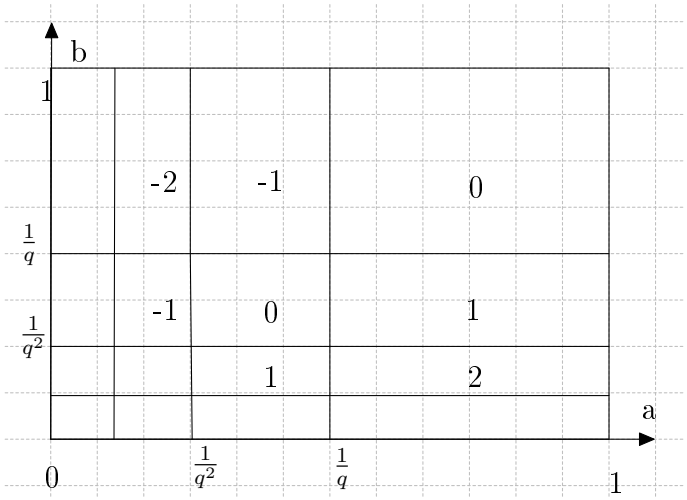
$$\alpha_\infty(f, s) = \frac{-q}{q^2 - 1} \sum_{i=1}^{h} q^{-|s - \deg r_i|}. \tag{9.21}$$

As a particular case, it is clear that the polynomials of degree 1 in x have exactly one infinite Laurent root. If one sets the skewness $s = \deg_t f$, then $\alpha_\infty(f, s) = -\frac{q}{q^2-1}$. As a second example, a degree-6 polynomial f over $\mathbb{F}_2$ which has infinite Laurent roots of degrees $3, 2, 1, 0, -1$ and -2 has $\alpha_\infty(f, 0) = -1.75$.

Example 9.2.20. A special class of polynomials are those corresponding to $\mathcal{C}_{\mathrm{a,b}}$ curves, which were proposed for the FFS in [Mat99]. If f is a $\mathcal{C}_{\mathrm{a,b}}$ polynomial and if we denote $\mathrm{a} = \deg_x f$ and $\mathrm{b} = \deg_t f_0 - \deg f_{\mathrm{a}}$, then for all $i \in [0, \mathrm{a} - 1]$ we have $\deg f_i < \deg f_{\mathrm{a}} + (\mathrm{a} - i)\frac{\mathrm{b}}{\mathrm{a}}$.

Suppose that a $\mathcal{C}_{\mathrm{a,b}}$ polynomial f had a Laurent root r. If $\deg r < \frac{\mathrm{b}}{\mathrm{a}}$, then $\max\{\deg f_i r^i \mid i \in [1, \mathrm{a}]\} < \deg f_0$. If $\deg r \geq \frac{\mathrm{a}}{\mathrm{b}}$ then $\deg f_{\mathrm{a}} r^{\mathrm{a}}$ dominates all the other terms of $f(r)$, so f has no Laurent roots. Hence, for any s, $\alpha_\infty(f, s) = 0$.

Figure 9.1: A domain of pairs (a, b) in lexicographical order having skewness s. We write the value of $\deg a - \deg b - s$ for each region where it is constant.



Constructing polynomials with many Laurent roots

One can easily check that, if a polynomial $f = \sum_i f_i x^i$ satisfies $\deg(f_d) = 0$, $\deg(f_{d-1}) = s$ for some $s > 0$ and $\deg(f_i) < (d - i)s$ for all $i \in [0, d - 2]$, then f has a Laurent root of degree s. This can be generalized to up to d roots.

For any edge $(v_i, i) \leftrightarrow (v_j, j)$ of the Newton polygon we call length the quantity $|i - j|$.

Proposition 9.2.21. *(Section II.6,[Neu99]) If $f \in \mathbb{F}_q[t][x]$ is a polynomial, each edge of length 1 in the Newton polygon corresponds to an infinite Laurent root for f.*

For example, the polynomial $f = x^7 + t^3x^6 + t^5x^5 + (t^6+1)x^4 + t^6x^3 + (t^5 + t + 1)x^2 + t^3x + 1$ has a Newton polygon with 7 edges of length 1 and therefore 7 infinite Laurent roots.

Note however that the converse of the Proposition 9.2.21 is false in general: a polynomial might have infinite Laurent roots which cannot be counted using the Newton polygon, e.g. for the polynomial f above, $f^1 := f(x+t^4)$ also has 7 infinite Laurent roots, although its Newton polygon has no edge of length 1.

9.3 Combining size, root and cancellation properties

The previous sections identified three elements which affect the sieve and defined associated measures: σ for the size property, α for the root property and α_∞ for the cancellation property. The list might be extended with other properties and the quantifying functions can be combined in different fashions in order to compare arbitrary polynomials. In this section we define two general purpose functions based on the three properties above and show their relevance through experimentation.

9.3.1 Adapting Murphy's $\mathbb{E}$ to the FFS

As a first function which compares arbitrary polynomials, we adapt Murphy's $\mathbb{E}$, already used for the NFS algorithm (Equation 5.7, [Mur99]). Heuristically, $\mathbb{E}$ uses Dickman's ρ to approximate the number of relations found by f and g on a sieving domain.

Definition 9.3.1. Let $f, g \in \mathbb{F}_q[t][x]$ be two irreducible polynomials, s an integer called skewness parameter, e a half integer called sieve parameter and β an integer called smoothness bound. Let $D(s,e)$ be the set of coprime pairs $(a,b) \in \mathbb{F}_q[t]^2$ such that $0 \leq \deg(a) \leq \lfloor e + \frac{s}{2} \rfloor$ and $0 \leq \deg(b) \leq \lfloor e - \frac{s}{2} \rfloor$. We define:

$$\mathbb{E}(f,g,s,e,\beta) = \sum_{(a,b)\in D(s,e)} \rho\left(\frac{\deg F(a,b) + \alpha(f)}{\beta}\right) \cdot \rho\left(\frac{\deg G(a,b) + \alpha(g)}{\beta}\right).$$

Unlike the situation in the NFS case, where $\mathbb{E}$ must be approximated by numerical methods, in the case of the FFS one can compute $\mathbb{E}$ in polynomial time with respect to $\deg(f)$, $\deg(g)$ and $e + |s|$. Note that ρ can be evaluated in polynomial time to any precision on the interval which is relevant in this formula.

We recall that we focus in this chapter on the case where the polynomial g is linear, as in [JL02]. More precisely, we assume that $g = g_1x + g_0$ is chosen with g_0 of much higher degree than g_1. In this case, the algorithm goes as follows:

1. compute the Laurent roots of f up to $\lfloor d(e + \frac{s}{2}) + \deg_t f \rfloor$ terms;

Table 9.1: Choosing the best skewness using $\mathbb{E}$. The parameters are set to $e = 24.5$ and $\beta = 22$.

s	-1	1	3	5	7
$10^{-5}E(f,g,s,e,\beta)$	2.54	3.31	3.46	2.88	2.12

2. for each $d_a \leq \lfloor e + \frac{s}{2} \rfloor$, $d_b \leq \lfloor e - \frac{s}{2} \rfloor$ and $i \in \mathbb{N}$, use Lemma 9.2.18 to compute the number $n(d_a, d_b, i)$ of pairs (a, b) such that $\deg(a) = d_a$, $\deg(b) = d_b$ and $\deg(F(a,b)) = i$;

3. compute

$$\sum_{d_a,d_b,i} n(d_a,d_b,i)\rho\left(\frac{i+\alpha(f)}{\beta}\right)\cdot\rho\left(\frac{d_b+\deg g_0+\alpha(g)}{\beta}\right). \tag{9.22}$$

One can use Murphy's $\mathbb{E}$ to choose the optimal skewness corresponding to a pair (f, g) of polynomials.

Example 9.3.2. Consider for instance the two polynomials used for the computation of the discrete logarithm in $GF(2^{619})$: $f = x^6+(t^2+t+1)x^5+(t^2+t)x+0\text{x}152\text{a}$ and $g = x - t^{104} - 0\text{x}6\text{dbb}$ written in hexadecimal[1] notation [BBD+12]. They used the smoothness bound 22 and most of the computations were done using special-Q's (q, r) with $\deg q = 25$. The pairs (a, b) considered for each special-Q were $(ia_0 + ja_1, ib_0 + jb_1)$ with (a_0, b_0) and (a_1, b_1) two pairs on the special-Q lattice and i, j were polynomials of degree at most 12. Hence, the pairs (a, b) considered were such that $\deg a + \deg b = \deg q + 24$, so $e = (\deg a + \deg b)/2 = 24.5$. Note that, if a and b have maximal degree on our set, the difference $\deg a - \deg b$ cannot be even. Table 9.1 shows that the best skewness value is 3.

Note though that in [BBD+12] one started by experimentally choosing the best skewness for polynomials of a given bound on the degrees. Then they selected polynomials which, for a given value of s, minimize the value of epsilon, the function that we define below.

An alternative to $\mathbb{E}$: Epsilon Recall that σ is the degree of the norm when no cancellation occurs, α is the degree gained due to the modular roots and α_∞ is the degree gained thanks to cancellations. It seems natural that their sum is the degree of a polynomial which has the same skewness probability as $F(a, b)$ for an "average" pair (a, b) on the sieving domain.

Definition 9.3.3. For a polynomial $f \in \mathbb{F}_q[t][x]$, a skewness parameter s and a sieve parameter e, we call epsilon the following average degree

$$\epsilon(f, s, e) = \alpha(f) + \alpha_\infty(f, s) + \sigma(f, s, e).$$

[1] Each polynomial ℓ of $\mathbb{F}_2[t]$, $\ell = \sum_i \ell_i t^i$ with $\ell_i \in \{0, 1\}$, is represented by base-16 notation of the integer $\sum_i \ell_i 2^i$.

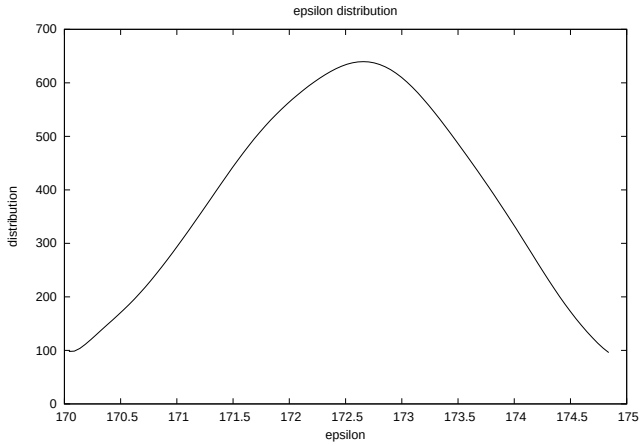


Figure 9.2: Distribution of epsilon on a sample of 20000 polynomials of $\mathbb{F}_2[t][x]$ of degree 6 in x and 12 in t.

Epsilon can be used to estimate the speedup of a polynomial with good properties. For example if the smoothness bound is 28 and two polynomials have the value of epsilon equal to 107 and 109 respectively, then we expect a speedup of $\rho(107/28)/\rho(109/28) \approx 1.19$.

Comparing ϵ and $\mathbb{E}$ Since the steps necessary in the computation of ϵ are equally used when evaluating $\mathbb{E}$, in practice epsilon is faster to compute than $\mathbb{E}$. The advantage of $\mathbb{E}$ is that it is more precise, but the experiments of the next section will show that epsilon is reliable enough.

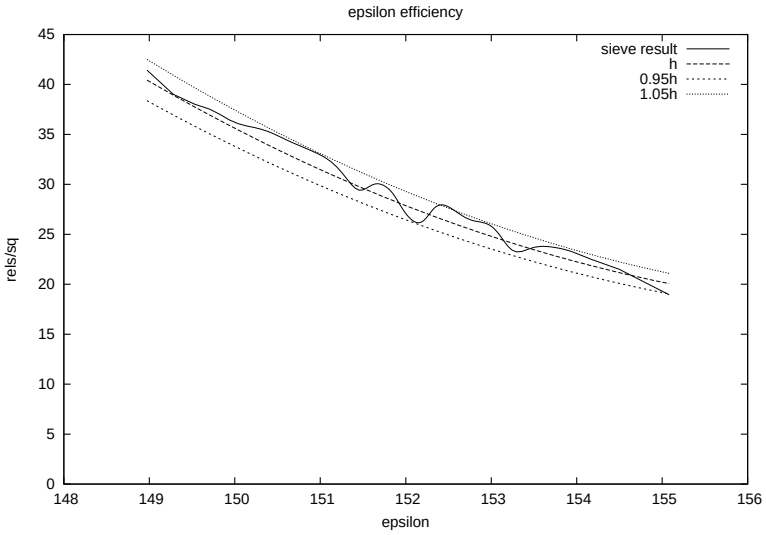
9.3.2 Experimental validation

The implementation used in the experiments is the one described in [DGV13], which is freely available at [BFG+09]. To our knowledge, no other press-a-button implementation of the FFS is publicly available. In addition, this implementation does relatively few modifications which could loose relations, making a theoretical study inexact.

The real-life efficiency of a polynomial is measured either by the number of relations per second, by the total number of relations, or as the average number of relations per special-Q. We kept the last one as a measure of efficiency since the software offers an option to reliably measure it and because it considers only the polynomial properties rather than the implementation quality.

Experiment 9.3.4. We selected a sample of polynomials f after evaluating epsilon for a range of polynomials considered one after another in lexicographical order starting from $x^6 + (t^2+t+1)x^5 + (t^3+t^2+t+1)x^3 + tx + t^{11}$. Note that the choice of the starting point guarantees that the polynomials considered have at least one infinite Laurent root. Since the distribution of epsilon was that of Figure 9.2,

Figure 9.3: Epsilon and sieve efficiency for the polynomials f in Experiment 9.3.4. The function h is a function of type $a + bx + c \log x$, with no special significance.



most of the polynomials tested had values of epsilon in a narrow interval. This lead us to select only one polynomial in each interval of length 0.01, to a total of 119 polynomials. Next we extended the sample with 60 polynomials starting from $x^6 + t^3x^5 + (t^5 + 1)x^4 + t^6x^3 + t^6 + 1$. For each polynomial f we associated a random monic linear polynomial g suited for FFS, having degree in t equal to 104. Indeed, as shown in Theorem 9.2.13 and in section 9.2.3 respectively, linear polynomial have the same values of α and α_∞ respectively.

We set the parameters as follows: I = J = 12, fbb0 = fbb1 = 22, lpb0 = lpb1 = 28, thresh0 = thresh1 = 100, sqside = 1. The polynomials q used in the special-Q technique were the first irreducible ones starting from t^{25}. We called the option "reliablenrels" which tests as many values of q as needed in order to obtain a measurement error of $\pm 3\%$ with a confidence level of 95%. The skewness parameter was set to $S = 3$ because, for the finite fields where the degree-6 polynomials are optimal, this is a sensible choice. Finally, the parameter sqt was set to 1 so that, for most special-Q's, the sieving domain was such that $\deg a \leq 26$ and $\deg b \leq 23$.

The results plotted in Figure 9.3 indicate that the sieve efficiency is not far from a strictly decreasing function of epsilon. To illustrates this, we plotted a decreasing function that fits our results, such that the relative error of our measurements is always less than 5%.

Finally, one can see that a sensible choice of the polynomial can save a factor 2 in the sieve time when compared to a bad choice.

Table 9.2: Influence of the linear polynomials g on the sieve efficiency. $f = \tilde{f} + f_0$ and $g = \tilde{g} + g_{00}$ with $\tilde{f}$ and $\tilde{g}$ given in section 9.3.3.

f_0	g_{00}	small factors of $\mathrm{Res}(f, g)$	efficiency (rels/sq)
0x19	0xb2	–	3.3
0x12	0xbf	–	3.4
0x12	0xb8	–	3.4
0x12	0xae	$t \cdot (t+1)$	3.5
0x12	0xa0	$t \cdot (t+1)$	3.5
0x12	0xbb	$t \cdot (t+1) \cdot (t^3+t^2+1)$	3.5
0x1e	0xbb	$t \cdot (t+1) \cdot (t^3+t+1) \cdot (t^6+\cdots)$	3.6

9.3.3 Correlation between f and g

A standard heuristic states that the probabilities of $F(a, b)$ and $G(a, b)$ to be smooth are independent, e.g. Murphy's $\mathbb{E}$ multiplies the two probabilities. The inexactness of this approximation could be called correlation property. According to Experiment 9.3.4, the correlation property has a small effect on the sieve, so that we bound ourselves to illustrate it by an example and a practical experiment.

Example 9.3.5. Let $f = x^2 - t^2(t+1)$, $g_1 = x - (t^2+t)^7$ and $g_2 = x - (t^2+t+1)^7$. Let F, G_1 and G_2 be the homogenizations of f, g_1 and g_2 respectively. Note that, for coprime pairs $(a, b) \in \mathbb{F}_q[t]$, $F(a, b)$ is divisible by t if and only if $a \equiv 0 \mod t$. For these pairs we have $G_1(a, b) \equiv 0 \mod t$ whereas $G_2(a, b) \not\equiv 0 \mod t$. In short, g_1 increases the number of doubly smooth pairs whereas g_2 that of pairs which are smooth on the rational or the algebraic side, but not on both.

If $\deg_x g = 1$, then every prime power ℓ^k of $\mathrm{Res}(f, g)$ implies a correlation between the events $\ell^k \mid F(a, b)$ and $\ell^k \mid G(a, b)$ on a domain of pairs (a, b). Table 9.2 summarizes an experiment in which we compared different pairs (f, g) with f having similar values of ϵ. We selected three polynomials f of the form $f = \tilde{f} + f_0$ with $\tilde{f} = x^6 + tx^5 + (t+1)x^4 + (t^2+t+1)x^3$ and we associated to each one a linear polynomial g of the form $g = \tilde{g} + g_{00}$ with $\tilde{g} = x - t^{104} - t^{14} + t^{13} + t^{11} + t^{10} + t^8$. Instead of imposing that $\mathrm{Res}(f, g)$ has an irreducible factor of degree 619 as in the previous experiment, we aimed to find polynomials g such that $\mathrm{Res}(f, g)$ has first no, then many, small factors. The experiment indicated that the correlation property explains part of the error observed in Experiment 9.3.4, bringing it close to 3%, which is equal to our measurement error.

Since it is easy to associate many linear polynomials g to a unique f and since linear polynomials have the same value of epsilon, it is interesting to select g such that $\mathrm{Res}(f, g)$ has many small factors. Nevertheless, f and g are chosen such that $\mathrm{Res}(f, g)$ has degree $d(\lfloor \frac{n}{d} \rfloor + 1)$ and we require an irreducible factor of degree n, leaving little room for additional factors. Moreover, when f imposes an extra factor to $\mathrm{Res}(f, g)$ (for example by having 1 projective and q affine roots modulo t), depending on the congruence of n modulo d, it can be impossible to choose g of optimal degree in t. See 9.5.2 for an example.

9.3.4 A sieving procedure for alpha

After we showed the relevance of epsilon, the polynomial selection comes to evaluating epsilon on a large set of polynomials. One can try various ranges of polynomials $f = \sum f_i x^i$, given by some degree bounds on their coefficients f_i, which optimize sigma and/or impose a number of Laurent roots, as shown in 9.2.3. The most time-consuming part of the computations, the evaluation of alpha, can be done on each range by a sieving procedure.

The idea is that, for each irreducible polynomial $\ell \in \mathbb{F}_q[t]$, we compute α_ℓ for all the residue polynomials f of $\mathbb{F}_q[t][x]$ modulo ℓ and then we update the values of α_ℓ for all the polynomials f in the range.

Let $d, e_0, \ldots, e_{d-1}$ and e_d be integers. We consider the range of the polynomials $f = \sum_{i=0}^{d} f_i x^i \in \mathbb{F}_q[t][x]$ such that for $i \in [0, d]$, $\deg_t f_i \leq e_i$. Call H the set of values taken by the tuple $(f_d, \ldots, f_2)$ and T those taken by (f_1, f_0). Let L be the set of irreducible polynomials up to a given bound. Let $k_{\max}$ be a parameter and let us suppose that, for all $\ell \in L$, all the roots $r \mod \ell^k$ with $\deg(\ell^k) \geq k_{\max}$ extend indefinitely.

For an irreducible polynomial ℓ, Algorithm 9.1 below computes $\alpha_\ell(f)$ for all f in the range and can be a steps to computing $\alpha(f)$ for the same range. We denote by residues(ℓ^k) the set of polynomials in $\mathbb{F}_q[t]$ of degree at most $k \deg \ell - 1$.

Algorithm 9.1 The alpha sieve

1: Initialize α_ℓ to a vector of value $\deg(\ell)/(\mathrm{N}(\ell) - 1)$
2: $k_0 \leftarrow \lceil k_{\max} / \deg \ell \rceil$
3: **for** $(f_d, f_{d-1}, \ldots, f_2)$ in H **do**
4: **for** k in $[1..k_0]$ and r in residues(ℓ^k) **do**
5: **for** (f_1, f_0) in T such that $f_1 r + f_0 \equiv -\sum_{i=2}^{d} f_i r^i \mod \ell^k$ **do**
6: $f \leftarrow \sum_{i=0}^{d} f_i x^i$
7: **if** $k < k_0$ **then**
8: $\alpha_\ell(f) \leftarrow \alpha_\ell(f) - \deg(\ell) \frac{\mathrm{N}(\ell)^{1-k}}{\mathrm{N}(\ell)+1}$
9: **else**
10: $\alpha_\ell(f) \leftarrow \alpha_\ell(f) - \deg(\ell) \frac{\mathrm{N}(\ell)^{2-k}}{\mathrm{N}(\ell)^2-1}$
11: **end if**
12: **end for**
13: **end for**
14: **end for**

The correctness of Algorithm 9.1 follows from Proposition 9.2.8. For a fixed value of $k_{\max}$, the complexity per polynomial is $O(1)$, as the most time-consuming steps are those in lines 8 and 10. For comparison, in the naive algorithm, for each polynomial, one needs to find the roots modulo ℓ, which takes a non-constant polynomial time in $d+\deg(\ell)$. In practice, Algorithm 9.1 showed to be much faster, as Paul Zimmermann used it to compute $\alpha_\ell(f)$, for all the irreducible polynomials ℓ with $\deg \ell \leq 6$, on the range of the 2^{48} monic polynomials $f \in \mathbb{F}_2[t][x]$ of degree 6 such that for $i \in [0, 6]$, $\deg_t f_i \leq 12 - 2i$.

9.4 Inseparable polynomials

9.4.1 Particularities of the inseparable polynomials

Despite the possibility of adding new technicalities, the inseparable polynomials have been preferred in two record computations [HSW+10], [HSST12]. Moreover, the Coppersmith algorithm, implemented in [Tho03], can be seen as a particular case of the FFS, using inseparable polynomials. In order to present the Coppersmith algorithm from this point of view and in order to compare inseparable polynomials to separable ones, we start with their definition, followed by their main properties.

Definition 9.4.1. An irreducible non-constant polynomial $f \in \mathbb{F}_q[t][x]$ is said inseparable if $f' = 0$, where f' denotes the derivative with respect to x.

For every inseparable polynomial f, there exists a power of the characteristic of $\mathbb{F}_q$, d, and a polynomial $\hat{f} \in \mathbb{F}_q[t][x]$ such that $f = \hat{f}(x^d)$ and $\hat{f}' \neq 0$. This simple property allows us to factor any irreducible polynomial ℓ in the function field of f in two steps. First we factor ℓ in the function field of $\hat{f}$, then we further factor every prime ideal $\mathfrak{l}$ of $\hat{f}$. The main advantage is that some prime ideal factorization algorithms work only for separable polynomials (for example Magma [BCP97] implements the function fields only in the case of separable polynomials). The factorization of the ideals $\mathfrak{l}$ of $\hat{f}$ in the function field of f is easy using the following result.

Proposition 9.4.2. *(Corollary X.1.8,[Lor96]) Let $p > 0$ be a prime and q and d two powers of p. Let $\hat{K}/\mathbb{F}_q(t)$ be a function field. Let $K/\hat{K}$ be an extension of polynomial $x^d - \theta_1$ with $\theta_1 \in \hat{K}$. Then every prime ideal $\mathfrak{l}$ of $\hat{K}$ decomposes as*

$$\mathfrak{l}\mathcal{O}_K = \mathfrak{L}^d \tag{9.23}$$

for a prime ideal $\mathfrak{L}$ such that $\mathfrak{L} \bigcap \mathcal{O}_{\hat{K}} = \mathfrak{l}$.

In the FFS algorithm, it is required to compute for each smooth element $a - b\theta$ of the function field of f, the valuation of every prime ideal $\mathfrak{L}$ in the factor base. For this, we start by factoring $(a - b\theta)^d$ in the integer ring of the function field $\hat{K}$ of $\hat{f}$:

$$(a - b\theta)^d \mathcal{O}_{\hat{K}} = (a^d - b^d\theta_1)\mathcal{O}_{\hat{K}} = \prod_i \mathfrak{l}_i^{e_i}$$

and then we obtain $(a - b\theta)\mathcal{O}_K = \prod_i \mathcal{L}_i^{e_i}$ where the $\mathfrak{L}_i$ are such that $\mathfrak{l}_i\mathcal{O}_K = \mathfrak{L}_i^d$.

9.4.2 Speed-up in the FFS due to the inseparability

Definition 9.4.3. Let f and g be two polynomials of $\mathbb{F}_q[t][x]$ such that $\mathrm{Res}(f, g)$ has an irreducible factor of degree n. Assume that $\deg_x g = 1$ and write $f = \hat{f}(x^d)$ for a separable polynomial $\hat{f}$ and some integer d which is either 1 or a power of $\mathrm{char}(\mathbb{F}_q)$. We call free relation any irreducible polynomial $\ell \in \mathbb{F}_q[t]$ such that $\ell \nmid \mathrm{Disc}(\hat{f}) f_d$ and $(f \mod \ell\mathbb{F}_q[t][x])$ splits into degree-1 factors.

Clearly each free relation of norm less than the smoothness bound creates an additive equation between the virtual logarithms of the ideals in the factor base.

The number of free relations is given by Chebotarev's Theorem as follows. First note that, due to Proposition 9.4.2, a polynomial ℓ is a free relation for f if and only if it is a free relation for $\hat{f}$. Then, the proportion of free relations among the irreducible polynomials is, according to Chebotarev's Theorem (see Corollary 3.1.3), asymptotically equal to the inverse of the cardinality of the Galois group of the splitting field of $\hat{f}$. Call N the number of monic irreducible polynomials in $\mathbb{F}_q[t]$ of degree less than the smoothness bound. Then, the number of free relations is:

$$\#\{\text{free relations}\} = \frac{N}{\#\operatorname{Gal}(\hat{f})}. \tag{9.24}$$

We compare this to the cardinality of the factor base. Since the cardinality of the rational side is N and f has as many ideals as $\hat{f}$, it is enough to evaluate the cardinality of the algebraic side. According to Chebotarev's Theorem, the number of pairs (ℓ, r) such that $f(r) \equiv 0 \mod \ell$, $\deg r < \deg \ell$ and $\deg \ell$ is less than the smoothness bound is χN where χ is the average number of roots of $\hat{f}$ fixed by the automorphisms of the splitting field of $\hat{f}$. It can be checked that each root of $\hat{f}$ is fixed by a fraction $1/\deg(\hat{f})$ of the automorphisms, so $\chi = 1$. Hence, asymptotically the factor base has $2N + o(N)$ elements. One could adapt the proof of Theorem 9.2.15 to obtain the sharper approximation of $2N + O(\sqrt{N})$.

Heuristically, $\operatorname{Gal} \hat{f}$ is the full symmetric group for all but a negligible set of polynomials $\hat{f}$, so most often we have $\#\operatorname{Gal}(\hat{f}) = \deg(\hat{f})!$. We list the results for $\deg f$ equal to 6 and 8 in Table 9.3. The case in which $d = 3$ and $\deg \hat{f} = 2$ brought a $\frac{4}{3}$-fold speedup in [HSW+10].

Coppersmith algorithm The case $d = 8$ and $\deg \hat{f} = 1$ corresponds to the Coppersmith algorithm. Indeed, since half of the relations are free relations, the sieve is accelerated by a factor of 2. Moreover, since $\deg(\hat{f}) = 1$, the free relations are particularly simple, linking exactly one element in the rational side to one element in the algebraic side of the factor base (Proposition 9.4.2). Therefore one can rewrite the relations using only the elements in the rational side, hence speeding up the linear algebra step by a factor of 4.

9.4.3 Root property of inseparable polynomials

Despite the fact that the inseparable polynomials are relatively few, being possible to exhaustively test them, it has its own interest to understand why inseparable polynomials have a bad root property and in particular, why the alpha value of many polynomials used in the Coppersmith algorithm is 2. Note that our proof that alpha converges covers only the case of separable polynomials. In this section we give some results on their root property.

First, the number of pairs (ℓ, r) with ℓ irreducible and r a polynomial of degree less than $\deg(\ell)$ such that $f(r) \equiv 0 \mod \ell$ has a narrower range of values than it does for the separable polynomials. Indeed, as shown by the following result, this number corresponds to the number of roots of $\hat{f}$. The bounds in Theorem 9.2.15,

Table 9.3: Number of free relations of a pair f, g with $f = \hat{f}(x^d)$, $\hat{f}$ separable and $\deg(g) = 1$. N is the number of irreducible monic polynomials of degree below the smoothness bound. The computations assume that $\#\operatorname{Gal}(\hat{f}) = (\deg \hat{f})!$.

$\operatorname{char}(\mathbb{F}_q)$	d	$\deg(\hat{f})$	#{factor base}	#{free relations}
any	1	6	$2N$	$N/720$
2	2	3	$2N$	$N/6$
3	3	2	$2N$	$N/2$
any	1	8	$2N$	$N/40320$
2	2	4	$2N$	$N/24$
2	4	2	$2N$	$N/2$
2	8	1	$2N$	N

when written explicitly, are narrower for polynomials of degree $\deg(\hat{f})$ than for those of degree $\deg f$. For example, if $\hat{f}$ is linear, this number is a constant.

Lemma 9.4.4. *Let $\hat{f} \in \mathbb{F}_q[t][x]$ be a polynomial, d a power of the characteristic of $\mathbb{F}_q$ and $f = \hat{f}(x^d)$. Let ℓ be an irreducible polynomial in $\mathbb{F}_q[t]$. Then there is a bijection between the sets $\{\hat{r} \in \mathbb{F}_q[t] \mid \deg \hat{r} < \deg \ell, \hat{f}(\hat{r}) \equiv 0 \mod \ell\}$ and $\{r \in \mathbb{F}_q[t] \mid \deg r < \deg \ell, f(r) \equiv 0 \mod \ell\}$.*

Proof. The non-zero residues of ℓ form a group of cardinality $\mathrm{N}(\ell) - 1$, which is coprime to q and hence to d. Therefore any root $\hat{r}$ accepts one and only one d^{th} root modulo ℓ. □

The second reason for having bad values of alpha is that most of the roots modulo irreducible polynomials ℓ do not lift to roots modulo ℓ^2. Recall the following classical result.

Lemma 9.4.5. *Let $\ell \in \mathbb{F}_q[t]$ be an irreducible polynomial. Write $(\mathbb{F}_q[t]/\langle \ell^2 \rangle)^*$ for the group of residues modulo ℓ^2 which are not divisible by ℓ. Put $U = \{e^{\mathrm{N}(\ell)} \mid e \in (\mathbb{F}_q[t]/\langle \ell^2 \rangle)^*\}$ and $V = \{1 + \ell w \mid \deg w < \deg \ell\}$. Then we have*

$$(\mathbb{F}_q[t]/\langle \ell^2 \rangle)^* \simeq U \times V.$$

The group U has order $\mathrm{N}(\ell) - 1$ which is coprime to d, so d^{th} roots always exist and are unique in U. On the other hand, in V, only the neutral element is a d-th power. As a consequence only a fraction $1/\#V = 1/\mathrm{N}(\ell)$ of the residues $\hat{r}$ modulo ℓ^2 can have d^{th} roots modulo ℓ^2. Let us make the heuristic assumption that the roots of $\hat{f}$ modulo ℓ^2 are random elements of $\mathbb{F}_q[t]/\langle \ell^2 \rangle$. Then only a small fraction of the roots of f lift modulo squares of irreducible polynomials and, for a non negligible fraction of polynomials f no root r modulo some irreducible polynomial ℓ lifts modulo ℓ^2.

Among the Coppersmith polynomials f, i.e., such that $\hat{f}$ is linear, many f are such that no modular root of f lifts modulo squares. Let us compute the value of alpha in this situation.

	$\alpha(f)$	$\alpha_\infty(f,s)$	$\sigma(f,s,e)$	$\epsilon(f,s,e)$	$E(f,g,s,e,\beta)$	efficiency
f_0	1.27	0	108.12	109.39	$1.82 \cdot 10^8$	15.2
f_1	−1.05	0	108.42	107.36	$2.10 \cdot 10^8$	18.8

Table 9.4: Coppermith polynomials for $\mathbb{F}_{2^{607}}$. The parameters in the table are $s = 7$, $e = 24.5$ and $\beta = 28$. The efficiency, measured in rels/sq, uses the parameters in Experiment 9.3.4.

Lemma 9.4.6. *Let $\hat{f}$ be a linear polynomial of $\mathbb{F}_q[t][x]$, d a power of the characteristic of $\mathbb{F}_q$ and put $f = \hat{f}(x^d)$. Assume that there is no pair of polynomials ℓ and r with ℓ irreducible and r of degree less than that of ℓ^2 such that $f(r) \equiv 0 \bmod \ell^2$. Then we have*

$$\alpha(f) = \frac{2}{q-1}.$$

Proof. By Lemma 9.4.4, for all ℓ, f has exactly $n_\ell = 1$ affine or projective roots modulo ℓ. By Corollary 9.2.9, for all irreducible polynomial ℓ we have

$$\alpha_\ell(f) = \deg \ell \left(\frac{1}{\mathrm{N}(\ell)-1} - \frac{1}{\mathrm{N}(\ell)} \frac{\mathrm{N}(\ell)}{\mathrm{N}(\ell)+1} \right) = 2 \frac{\deg \ell}{q^{2\deg \ell}-1}.$$

Hence we obtain $\alpha(f) = 2\left(\sum_{k \geq 1} \frac{kI_k}{q^{2k}-1} \right)$ with I_k the number of irreducible monic polynomials of degree k in $\mathbb{F}_q[t]$. The sum in the parenthesis was computed in Proposition 9.2.13 and equals $1/(q-1)$. This completes the calculations. □

9.5 Applications to some examples in the literature

9.5.1 Thomé's record using the Coppersmith algorithm

Thomé [Tho03] solved the discrete logarithm problem in $\mathbb{F}_{2^{607}}$ using the Coppersmith algorithm. Following this algorithm, one sets $g = x - t^{152}$ and $f = x^4 + t\lambda$ for a polynomial $\lambda \in \mathbb{F}_q[t]$ such that $t^{607} + \lambda$ is irreducible. The polynomial $\lambda_0 = t^9 + t^7 + t^6 + t^3 + t + 1$ used by Thomé minimizes the degree of λ. If one searches for an alternative, it is necessary to increase $\deg_t f$, but this is possible without affecting much the size property. Indeed, the sensible choice is to set the skewness s to 7, so sigma does not vary much if one increases $\deg f_0$. By testing the polynomials λ with $\deg \lambda \leq 18$, we determined that the best alpha corresponds to $f_1 = x^4 + t(t^{16} + t^{12} + t^{11} + t^7 + t^4 + 1)$. We compare the two polynomials in Table 9.4 using the functions defined in this article as well as the sieve efficiency measured with the implementation of [DGV13] and the parameters in Experiment 9.3.4. We conclude that the new polynomial for the Coppersmith algorithm obtains 23% more relations for the same sieved domain.

f, g	$\alpha(f)$	$\alpha_\infty(f, s)$	$\sigma(f, s, e)$	$\epsilon(f, s, e)$	$E(f, g, s, e, \beta)$	efficiency
f_2, g_2	2.15	0	122.33	124.46	$8.54 \cdot 10^8$	66.0
f_3, g_3	−0.24	0	123.66	123.36	$8.64 \cdot 10^8$	73.8
f_4, g_4	−0.10	0	123.66	123.42	$9.49 \cdot 10^8$	76.0

Table 9.5: Classical FFS polynomials for $\mathbb{F}_{2^{607}}$. The parameters are $s = 1$, $e = 24.5$, $\beta = 28$. The efficiency, measured in rels/sq, uses the parameters in Experiment 9.3.4.

9.5.2 Joux-Lercier's implementation of the classical variant of the FFS

Joux and Lercier [JL02, JL07] considered the fields $\mathbb{F}_{2^n}$ with $n = 521$, 607 and 613. For $n = 607$ they set $f_2 = x^5 + x + t^2 + 1$ and $g_2 = (t^{121} + t^8 + t^7 + t^5 + t^4 + 1)x + 1$. If one searches for an alternative, the sensible choice is to improve the root property without changing the size property. Since $\deg_t f_2 = 2$ we tested all the polynomials whose degree in t is 1 or 2.

Experiment 9.5.1. There are 2^{18} polynomials f such that $\deg_t f \leq 2$, out of which 2^{12} have $\deg_t f \leq 1$. There were 1776 irreducible polynomials with $\deg_t f \leq 1$ whose alpha is below 3. There were 650 irreducible polynomials f, with $\deg_t f \leq 2$, whose alpha is negative and such that the partial sum of alpha up to degree 6 is less than 0.5.

The best 10 values for ϵ with skewness $s = 0$ and sieve parameter $e = 24.5$ were all obtained for polynomials f with $\deg_t f = 2$. The best value was that of $f_3 = (t^2 + t)x^5 + (t^2 + t + 1)x^4 + (t + 1)x^3 + t^2x^2 + t^2x + t^2$. We could not associate a linear polynomial g with $\deg_t g = 121$ because $\mathrm{Res}(f, g)$ is always divisible by t (see 9.3.3 for more details), hence we took $g_3 = x + t^{122} + t^{13} + t^{11} + t^6 + t^5 + t^3 + t^2$. The best f for which we could select a linear polynomial g of degree 121 was $f_4 = (t^2 + t + 1)x^5 + (t^2 + t + 1)x^4 + x^3 + (t^2 + t + 1)x^2 + (t^2 + t + 1)x + t^2 + t$, for which we took $g_4 = x + t^{121} + t^{12} + t^{11} + t^8 + t^6 + t^2 + 1$. In Table 9.5 we compare (f_3, g_3) and (f_4, g_4) to (f_2, g_2). Note also that all the polynomials f tested have a small genus, which could explain the small variance of alpha when $\deg_t f \leq 2$.

We conclude that

- the new polynomial obtains 15% more relations for the same sieved domain than the polynomial used by Joux and Lercier.

- the new polynomial is 2.5 times more effective than Thomé's polynomial after we take into acount the difference between FFS and the Coppersmith algorithm. Indeed, with the new polynomial we obtained 5 times more relations than with Thomé's polynomial. We divide this by 2, since Coppersmith's algorithm requires 2 times less relations, as shown in Table 9.3.

9.5.3 Joux-Lercier's two rational side variant

In this chapter we focussed on the classical variant of FFS. A short comparison with the Two rational side variant in Section 5.4 is as follows. First, the polynomials which are expected to be smooth, $F(a,b)$ for the classical variant and the x and y-sides in Equation (5.14) for the two rational side variant, have the same degree as shown by Equations (5.21) and (9.1).

Let us next consider the root property. We rename t the variable y in the Two rational side variant. We call $f = \gamma_2(x) - t$ and $g = x - \gamma_1(t)$. The considerations due to the size property impose that f has the same degree in the two variants. One can check that the x and, respectively y-side, of a pair (a,b) in the algorithm are $\mathrm{Res}_x(g, a-bx)$ and respectively $\mathrm{Res}_t(f, a-bx)$. Since the y-side, $\mathrm{Res}_x(g, a-bx)$ corresponds to the norm of the linear polynomial in the classical version, no root property can be imposed on this side. The x-side, $\mathrm{Res}_t(f, a-bx)$ cannot be easily be seen as a norm of an element of a function field over $\mathbb{F}_q(t)$, so our analysis does not apply. Nevertheless, the number of choices of f is very limited. For example, there are 2^6 possible choices for f such that $\deg_x f = 6$.

9.5.4 Records on pairing-friendly curves

The fields $\mathbb{F}_{3^{6n}}$ are of particular interest in cryptography as one can break the cryptosystems which use pairing-friendly curves over $\mathbb{F}_{3^n}$ by solving the discrete logarithm problem in $\mathbb{F}_{3^{6n}}$. The recently proposed algorithms [Jou13a],[Jou13b],[GGMZ13] and [BGJT13] proved to be very fast for the fields of composite degree. It rendered FFS obsolete in this case and drastically reduced the security of these curves, as was noted in [AMORH13]. This section is also interesting for illustrating the behaviour separable polynomials.

These fields allow us to run the FFS with a base field $\mathbb{F}_{3^d}$ with $d = 2, 3$ or 6. Hence we collect coprime pairs (a,b) of polynomials in $\mathbb{F}_{3^d}[t]$ such that both $F(a,b)$ and $G(a,b)$ factor into small degree polynomials of $\mathbb{F}_{3^d}[t]$. The Galois variant [JL06] consists in choosing the polynomials f and g to have their coefficients in $\mathbb{F}_3[t]$ rather than $\mathbb{F}_{3^d}[t]$. Its main advantage is that, due to Galois properties, the factor base is reduced by a factor of d.

Hayashi *et al.* [HSW+10] used the base field $\mathbb{F}_{3^6}$ and the polynomial $f_i = x^6 + t$ to break curves over $\mathbb{F}_{3^{71}}$. Two years later, Hayashi *et al.* [HSST12] used the base field $\mathbb{F}_{3^3}$ and again $f_i = x^6 + t$ to break cryptosystems over $\mathbb{F}_{3^{97}}$.

Since the polynomial f_i is inseparable, as explained in 9.4.2, one quarter of the relations collected by the FFS are free. This roughly translates into a 4/3-fold speedup with respect to the separable polynomials having the same sieve efficiency. Note that f_i has the best epsilon among the 486 inseparable polynomials f in $\mathbb{F}_3[t][x]$ with $\deg_t f \leq 1$ and $\deg_x f = 6$.

A better choice can be only a separable polynomial with a better efficiency. Since the efficiency of a polynomial depends on the base field of the factor base, we distinguish the case of $\mathbb{F}_3$ from the case of $\mathbb{F}_{3^d}$ with $d = 2, 3$ or 6.

Experiment 9.5.2. Since $\deg_t(f_i) = 1$ we can use any of the $8 \cdot 3^{12}$ polynomials f in $\mathbb{F}_3[t][x]$ such that $\deg_t f \leq 1$, without changing the size property. The best alpha with respect to $\mathbb{F}_3$ corresponded to $f_s = tx^6 - tx^4 + (-t+1)x^3 + (t-1)x + t$.

f, g	$\alpha(f)$	$\alpha_\infty(f, s)$	$\sigma(f, s, e)$	$\epsilon(f, s, e)$	$E(f, g, s, e, \beta)$	efficiency
f_i	1.33	0	94.00	95.33	$1.03 \cdot 10^8$	14.6
f_s	0.29	0	94.75	95.04	$1.23 \cdot 10^8$	17.0
$f_{s'}$	−3.67	0	96.75	93.03	$1.61 \cdot 10^8$	21.3

Table 9.6: Polynomials f for fields of characteristic 3. The last column was obtained using same software as in Experiment 9.3.4 and with parameters fbb0=fbb1=14, lpb0=lpb1=17, S=1 and $q0 = t^{15}$.

f	$\overline{\alpha}(f, \mathbb{F}_3)$	$\overline{\alpha}(f, \mathbb{F}_{3^2})$	$\overline{\alpha}(f, \mathbb{F}_{3^3})$	$\overline{\alpha}(f, \mathbb{F}_{3^6})$
f_i	2.11	0.35	0.53	0.03
f_s	0.46	0.16	0.21	0.08

Table 9.7: The values of alpha with respect to different base field. The notation $\overline{\alpha}(f, \mathbb{F}_q)$ denotes $\log(q)/\log(2) \cdot \alpha(f)$ with alpha corresponding to the field $\mathbb{F}_q$.

Since alpha has a small variance on the polynomials tested in Experiment 9.5.2, we also consider the separable polynomial $f_{s'} = x^6 - x^2 + (t^8 + t^6 - t^4 + t^2 + 1)$, which is well suited when the skewness parameter is set to 1. Table 9.6 compares f_s and $f_{s'}$ to f_i for randomly chosen linear polynomials.

In the case when the base field is $\mathbb{F}_{3^6}$ and $\mathbb{F}_{3^3}$ the evaluation of alpha is slower, with a factor of 200 compared to the case of $\mathbb{F}_{3^6}$. Note first that the polynomial f_s, whose root property over $\mathbb{F}_3$ is better than that of f_i, has a poorer value of alpha when the base field is $\mathbb{F}_{3^6}$. In Table 9.7 we use $\frac{\log q}{\log 2} \times \alpha$ as an alternative of alpha which allows us to compare polynomials f with coefficients in different rings $\mathbb{F}_q[t]$. The values of alpha are approximated by considering only the contribution of at most 1000 irreducible polynomials in $\mathbb{F}_{3^d}$ with $d = 1, 2, 3$ or 6. Note that the values of $\frac{\log q}{\log 2} \times \alpha$ are close to each other when $q = 3^6$. This opens the question of how does the distribution of alpha evolve when we compute it with respect to a factor base in $\mathbb{F}_{3^6}[t]$ but for which the polynomials f are in $\mathbb{F}_3[t][x]$.

9.6 Conclusion

Improving on Joux and Lercier's method of polynomial selection [JL02], we noted that a unique polynomial f can be used to solve the discrete logarithm problem on a range of inputs. Since the selection of f can be seen as a precomputation, we developed a series of functions which compare arbitrary polynomials and which are much faster than directly testing the sieve efficiency. In particular we obtained a sieving procedure for computing alpha, the function which measures the root property and we defined a function for measuring the cancellation property.

The case of inseparable polynomials was of particular interest as it has no equivalent notion in the NFS world. We showed that inseparable polynomials have the advantage of a large number of free relations, but most of the inseparable polynomials have a bad root property. The last section applied the new functions to

some records in the literature. We concluded that the inseparable polynomails offer a better efficiency for the examples we considered. In particular, we saw an example where Coppersmith's algorithm (FFS with totaly inseparable polynomials) is 2.5 times slower than FFS. Since our experiments did not consider the case of values of q other than 2 and 3, we cannot decide if the separable polynomials are more effective for larger base fields, e.g. $q = 3^6$.

Chapter 10

A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic

We conclude this part with an algorithm that is not a variant of FFS although during its genesis it originally was. This comes in continuation to recent progress made in 2013 by Joux [Jou13a], Joux [Jou13b] and Göloğlu and others [GGMZ13]. In a joint work with Joux, Gaudry and Thomé [BGJT13] we finally showed that under reasonable heuristics one can solve the DLP in finite fields of small characteristic in a quasi-polynomial time. As a nice bonus, this algorithm no longer makes use of function field theory and is not more complicated than the algorithms that we called basic in Chapter 5.
We organize the chapter as follows. We start with a short recapitulation of the recent progress on the topic. We then present the setting and we show how to compute the discrete logarithms of the factor base in polynomial time. We then state the main result and show how to perform each descent step. We conclude with arguments in support of our assumptions and we measure the consequences of a slightly stronger assumption.

10.1 Recent DLP progress

The new wave of improvements started at the end of 2012 when Joux [Jou13a] proposed a new technique for the relation collection stage of FFS. In the Two rational side variant of FFS, presented in Section 5.4, after setting γ_1, γ_2 and D in the polynomial selection stage, we collect univariate polynomials A and B such that $A\left(\gamma_2(x)\right)x + B\left(\gamma_2(x)\right)$, the x-side, and $A(y)\gamma_1(y) + B(y)$, the y-side, are both D-smooth. The basic idea now is to use the maps which send the factor base to itself and which send a pair (A, B) with both sides smooth to another pair (A', B') with at least one of the x and the y-sides smooth. Note that this idea was already used in the so-called Galois improvements in [JL06, Sec. 2.3] and [JLSV06, Sec. 4.3]. Joux sets $\gamma_1(y) = y^{d_2}$ for some parameter d_2 (notation conformal to [Jou13a]) and uses a map called amplification: $\big(A(y), B(y)\big) \mapsto \big(a^{-d_2}A(y/a), B(y/a)\big)$. Note that

if we call T the y-side of the pair (A, B), then the y-side of $\left(a^{-d_2}A(y/a), B(y/a)\right)$ equals $T(y/a)$:

$$[a^{-d_2}A(y/a)] \cdot y^{d_2} + B(y/a) = [A(y)y^{d_2} + B(y)](y/a). \tag{10.1}$$

Hence, if (A, B) is a pair such that the y-side is D-smooth, then we obtain for free $q-2$ other pairs with the same property. When collecting relations for FFS, we start by trying random values of A and B until the y-side is D-smooth. Next we obtain $q-2$ more pairs (A, B) by amplification and, for each pair (A, B), we test the D-smoothness of the x-side. Joux further proposed the so-called Kummer variant, which suits those values of k where we can set $\gamma_1(y) = y^{d_2}$ and $\gamma_2(x) = x^{d_1}/\alpha$ for some parameters d_1, d_2 and α. For this choice of γ_1 and γ_2, both the x and the y-sides remain D-smooth when we amplify a doubly-smooth pair (A, B).

Göloğlu, Granger, McGuire and Zumbrägel [GGMZ13] further improved the case of fields $\mathbb{F}_{q^k}$ when q is a power of 2. The factor base is formed of degree 1 polynomials, which corresponds to taking parameter $D = 1$ in FFS. The sieving space are the polynomial pairs $(A, B) = (a + x, bx + c)$. The most surprising result is that the relation collection stage takes a polynomial time when $q \approx k$ and k has a constant divisor $k' \geq 3$. Indeed, they set $\gamma_2(x) = x^q$ and they impose that the degree of $\gamma_1(y)$ is bounded by a constant; let us focus on the case $\gamma_1(y) = h_{01}y + h_{00}$. Since for all $a \in \mathbb{F}_{q^{k'}}$ we have $(y + a)^q = x + a^q$ we can restrict the factor base to the polynomials in x. The polynomial selection consists in choosing $h_{00}, h_{01} \in \mathbb{F}_{q^{k'}}$ such that $x^q - (h_{01}x + h_{00})$ has an irreducible factor of degree k/k'. In the relation collection stage, the degree of the y-side is bounded by a constant, so it has a constant probability to be 1-smooth. On the other hand, for any pair $(A, B) = (x + a, bx + c)$, the x-side is a polynomial of a very special form: $x^{q+1} + ax^q + bx + c$. By Theorem 1 in [HK10] a polynomial of this form has a probability of approximatively $1/q^3$ to be 1-smooth instead of $\rho(q + 1) \approx q^{-q(1+o(1))}$ as expected for a random polynomial of the same degree. In particular, the analysis done in Chapter 9 does not apply here because the polynomials A and B are linear whereas we assumed $\deg A, \deg B \to \infty$.

Joux [Jou13b] independently obtains similar results, but he invents a new descent technique which leads to a $L(1/4 + o(1))$ algorithm. As in the previous algorithm he uses a factor base composed of degree 1 polynomials in x. As in the previous algorithm the best case is when $q \approx k$ and k has a constant divisor k' which here can be equal to 2. The polynomial selection stage searches for $h_{00}, h_{01}, h_{10}, h_{11} \in \mathbb{F}_{q^{k'}}$ such that the numerator of $x^q - \frac{h_{01}x+h_{00}}{h_{11}x+h_{10}}$ has a degree k/k' irreducible factor. But Joux's algorithm is not a variant of FFS since no polynomials γ_1 and γ_2 are made explicit. Moreover, the relation collection stage in polynomial time is achieved by a different technique called systematic equations, that we describe in detail in Section 10.4. The descent is done in three steps:

- Smoothing, also called ignition of the descent in Section 6.6.2 or continued fraction method in [AMORH13]. The smoothing bound is tuned so that the smoothness probability is $L(1/4)^{-1}$.
- Classical descent by special-Q. Here we make use of two polynomials γ_1 and γ_2 as in the two rational side FFS, but we do not make them explicit. This allows to use the same technique as in FFS.

- Bilinear system descent making use of Gröbner basis. As in the special-Q descent we express the logarithm of a degree m polynomial with respect to degree $m-d$ polynomials with $d = \log_q\big(L(1/4)\big)$. We end the descent at the polynomials of degree 2. Descending from degree 2 to degree 1 cannot be done by Gröbner basis, but Joux proposed a different technique which will be generalized in Section 10.7.

10.2 Our results

In the present chapter, we present a new discrete logarithm algorithm, in the same vein as in [Jou13b] that uses an asymptotically more efficient descent approach. The main result gives a *quasi-polynomial* heuristic complexity for the DLP in finite field of small characteristic. By quasi-polynomial, we mean a complexity of type $n^{O(\log n)}$ where n is the bit-size of the cardinality of the finite field. Such a complexity is smaller than any $L(\varepsilon)$ for $\epsilon > 0$. It remains super-polynomial in the size of the input, but offers a major asymptotic improvement compared to $L(1/4 + o(1))$.

The key features of our algorithm are the following.

- We keep the field representation and the systematic equations of [Jou13b].
- The algorithmic building blocks are elementary. In particular, we avoid the use of Gröbner basis algorithms.
- The complexity result relies on three key heuristics: the existence of a polynomial representation of the appropriate form; the fact that the smoothness probabilities of some non-uniformly distributed polynomials are similar to the probabilities for uniformly random polynomials of the same degree; and the linear independence of some finite field elements related to the action of $\mathrm{PGL}_2(\mathbb{F}_q)$.

The heuristics are very close to the ones used in [Jou13b] and we will complement them with a discussion in Section 10.8.

Although we insist on the case of finite fields of very small characteristic, where quasi-polynomial complexity is obtained compared to the previous $L(1/4 + o(1))$, our new algorithm improves the complexity of discrete logarithm computations in the whole range of application for FFS except for the middle prime case, i.e., $q = L_Q(1/3)$. More precisely, in finite fields of the form $\mathbb{F}_Q$, where $Q = q^k$ and q grows as $L_Q(\alpha)$, the complexity becomes $L_Q(\alpha + o(1))$. As a consequence, our algorithm is asymptotically faster than the Function Field Sieve algorithm in almost all the range previously covered by this algorithm. Whenever $\alpha < 1/3$, our new algorithm offers the smallest complexity. For the limiting case $L(1/3, c)$, the function field sieve remains more efficient for small values of c, and the number field sieve is better for large values of c (see [JLSV06]).

10.3 Setting

We start by describing the setting in which our algorithm applies. We make the same construction as in [Jou13b] but we apply it to any field $\mathbb{F}_{q^k}$ with q smaller than $L_{q^k}(1/3, \cdot)$.

Let us consider first the case $k \leq q + \delta$ for a constant δ. We will give later heuristic arguments that one can always use $\delta = 2$ but any constant is enough for our result. We try random polynomials $h_0(X)$ and $h_1(X)$ of degree at most δ in $\mathbb{F}_{q^2}$ until $h_1 X^q - h_0$ has a degree k irreducible factor $\varphi(X)$. We represent $K = \mathbb{F}_{q^{2k}}$ as

$$K = \mathbb{F}_{q^2}[X]/\varphi(X) \cong \mathbb{F}_{q^{2k}}.$$

Let us next consider the case $k > q + \delta$. Then we replace q by a suitable q' and compute discrete logarithms in $\mathbb{F}_{(q')^{2k}}$, which coincide with discrete logarithms in $\mathbb{F}_{q^{2k}}$. We will see in Section 10.6 that this has little impact on the complexity.

10.4 Logarithms of the factor base

We call x a root of φ in $\mathbb{F}_{q^{2k}}$ and note the following equality:

$$x^q = \frac{h_0(x)}{h_1(x)}. \tag{10.2}$$

We define the factor base as the set of elements in $\mathbb{F}_{q^{2k}}$ which can be written as monic degree 1 polynomials in x to which we add $h_1(x)$

$$\mathcal{F} = \left\{x + a \mid a \in \mathbb{F}_{q^2}\right\} \bigcup \{h_1(x)\}. \tag{10.3}$$

We compute the discrete logs of elements in the factor base as follows. Consider the *systematic equation*

$$Y^q - Y = \prod_{(\alpha:\beta)\in\mathbb{P}^1(\mathbb{F}_q)} (\beta Y - \alpha), \tag{10.4}$$

where the representatives of $\mathbb{P}^1(\mathbb{F}_q)$ are chosen so that the right-hand side is indeed monic. For all quadruples $(a, b, c, d) \in \mathbb{F}_{q^2}$ with $(a, b) \neq (0, 0)$ and $(c, d) \neq (0, 0)$ we substitute $\frac{ax+b}{cx+d}$ to Y in the systematic equation and we multiply by $(cx + d)^{q+1}$:

$$(ax+b)^q(cx+d) - (ax+b)(cx+d)^q = \prod_{(\alpha:\beta)\in\mathbb{P}^1(\mathbb{F}_q)} \left(\beta(ax + b) - \alpha(cx + d)\right). \tag{10.5}$$

Due to Equation (10.2), we can also write the left hand member as follows:

$$\left(\tilde{a}\frac{h_0}{h_1} + \tilde{b}\right)(cx + d) - \left(\tilde{c}\frac{h_0}{h_1} + \tilde{d}\right)(ax + b) = (ax + b)^q(cx + d) - (ax + b)(cx + d)^q, \tag{10.6}$$

where $\tilde{a} = a^q$, $\tilde{b} = b^q$, $\tilde{c} = c^q$ and $\tilde{d} = d^q$.

Hence, to each quadruple (a, b, c, d) one obtains:

$$\left(\tilde{a}\frac{h_0}{h_1} + \tilde{b}\right)(cx + d) - \left(\tilde{c}\frac{h_0}{h_1} + \tilde{d}\right)(ax + b) = \prod_{(\alpha:\beta)\in\mathbb{P}^1(\mathbb{F}_q)} \left(\beta(ax + b) - \alpha(cx + d)\right). \tag{10.7}$$

We start the algorithm by collecting $\#\mathcal{F}$ such relations for which the left hand member is 1-smooth and writing down corresponding equations in terms of the discrete logarithms. If the obtained linear system can be solved, we obtain the discrete logarithms of the elements in the factor base.

How many distinct equations are there? Let us associate to each quadruple (a, b, c, d) the matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Note first that quadruples with singular matrices give a trivial relation in Equation (10.5). One can check that two quadruples give the same right hand member in Equation (10.5) if and only if their associated matrices have the same class in $\mathcal{P}_q := \mathrm{PGL}(2, \mathbb{F}_{q^2})/\mathrm{PGL}(2, \mathbb{F}_q)$. Since, for all fields κ, $\mathrm{PGL}(2, \kappa)$ has $\#\kappa^3 - \#\kappa$ elements, there are $q^3 + q$ quadruples to consider.

The numerator of $(\tilde{a}\frac{h_0}{h_1} + \tilde{b})(cX + d) - (\tilde{c}\frac{h_0}{h_1} + \tilde{d})(aX + b)$, call it $\mathcal{L}$, has a degree less than or equal to $\delta + 1$. Under the heuristic assumption that it has the same smoothness probability as a random polynomial of degree $\delta + 1$, there are approximatively $(q^3 + q)/(\delta + 1)!$ quadruples (a, b, c, d) for which $\mathcal{L}$ is 1-smooth.

Since the system has q^2 unknowns and $\Theta(q^3)$ equations, we make the heuristic assumption that its associated matrix has full rank. One completes the computation by solving the linear system. This can be done either with sparse matrix techniques (see Section 6.3.4) or with Gaussian elimination, depending on the exponent ω of the matrix multiplication complexity.[1] If one uses sparse matrix techniques, one has a complexity of $O(q^5)$. Indeed, the size of the matrix is q^2–we use $\#\mathcal{F} \approx q^2$ rows of the matrix in order to obtain a square matrix. The number of non-zero entries per row is at most $q + 3 + \delta$: one for $h_1(x)$, $\delta + 1$ factors in $\mathcal{L}$, and $q + 1$ factors in the right hand member of Equation (10.7).

10.5 Main result

Proposition 10.5.1. *Under plausible heuristics, there exists an algorithm that takes as input a field K as above, and a polynomial $P(X)$ of degree $D \leq k$ over $\mathbb{F}_{q^2}$ and returns a linear relation between the discrete logarithm of P in K and the discrete logarithms of $O(q^2 D)$ polynomials of degree less than $\max(2, \lceil D/2 \rceil)$ in time bounded by a polynomial in q and D.*

The algorithm is given in the next section. We first show how to use this as a building block for a complete discrete logarithm algorithm.

Let P be an element of K for which we want to compute the discrete logarithm. We can assume that P is represented by a polynomial of degree less than k over $\mathbb{F}_{q^2}$. We start by applying the algorithm of Proposition 10.5.1 to P. We obtain a relation of the form

$$\log P = \sum e_i \log P_i,$$

[1] The idea of dense matrix techniques was suggested to us by Dan Bernstein.

where the sum has at most $\kappa q^2 k$ terms for a constant κ and the P_i's have degree less than $\lceil \deg P/2 \rceil$. Then, we apply recursively the algorithm to the P_i's, thus creating a descent procedure where at each step, a given element P is expressed as a product of elements, whose degree is at most half the degree of P (rounded up) and the arity of the descent tree is in $O(q^2 k)$.

At the end of the process, the logarithm of P is expressed as a linear combination of the logarithms of linear polynomials. In Section 10.4 we showed that these can be computed in polynomial time in q. So we are left with the complexity analysis of the descent process.

Each internal node of the descent tree corresponds to one application of the algorithm of Proposition 10.5.1, therefore each internal node has a cost which is bounded by a polynomial in q and k. The total cost of the descent is therefore bounded by the number of nodes in the descent tree times a polynomial in q and k. The depth of the descent tree is in $O(\log k)$. At the i-th depth-level, we have at most $(\kappa q^2 k)^i$ polynomials. Therefore, the number of internal nodes is bounded by $(q^2 k)^{O(\log k)}$. Since any polynomial in q and k is absorbed in the $O()$ notation in the exponent, we obtain the following result.

Theorem 10.5.2. *Assuming the same heuristics as in Proposition 10.5.1, any discrete logarithm in K can be computed in time bounded by*

$$\max(q,k)^{O(\log k)}.$$

10.6 Consequences for various ranges of parameters

Case where q is polynomial in the input size. Assume $q = (\log Q)^{O(1)}$. If $k \leq q + \delta$ we can apply directly the algorithm. Since $k \log q = \log Q$ and since we are in the case $q = \log Q^{O(1)}$ we obtain $\max(q,k)^{\log k} \leq 2^{O\left((\log\log Q)^2\right)}$.

If $k > q + \delta$ we put $q' = q^{\lceil \log_q k \rceil}$ and apply the algorithm to $\mathbb{F}_{q'^{2k}}$. Note that $q' \leq kq \leq (\log Q)^{O(1)}$. Then the complexity is $\max(k,q')^{O(\log k)}$. We obtain a complexity $\max(q,k)^{O(\log k)} \leq 2^{O\left((\log\log Q)^2\right)}$ with a constant hidden in $O()$ which is larger than in the previous case. It is worth emphasizing that the case $K = \mathbb{F}_{2^n}$ for a prime n corresponds to this case.

Case where $q = L_{q^k}(\alpha)$. If the characteristic of the base field is not so small compared to the extension degree, the complexity of our algorithm does not keep its nice quasi-polynomial form. However, in almost the whole range of applicability of the Function Field Sieve algorithm, our algorithm is asymptotically better than FFS.

With our hypothesis, $\log q = (\log Q)^\alpha (\log\log Q)^{1-\alpha}$ and therefore $k = \log Q / \log q = O((\log Q / \log\log Q)^{1-\alpha})$. In particular we have $k \leq q + \delta$.

The complexity of Theorem 10.5.2, takes the form $q^{O(\log k)} = L_Q(\alpha)^{O(\log\log Q)}$. This is smaller than $L_Q(\alpha')$ for any $\alpha' > \alpha$.

Hence, for any $\alpha < 1/3$, our algorithm is faster than the previously best known algorithm, namely FFS and its variants.

10.7 Algorithm for one descent step

Proof of Proposition 10.5.1. To decompose the logarithm of $P(X)$, we are going to decompose *simultaneously* all the translates of $P(X)$ by a constant in $\mathbb{F}_{q^2}$.

To each element $m = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ of $\mathcal{P}_q$, we associate the Equation (E_m) obtained by substituting $m \cdot P = \frac{aP+b}{cP+d}$ in place of Y in the systematic equation (Equation (10.4)). Clearing denominators with a multiplication by $(cP+d)^{q+1}$, it becomes:

$$\begin{aligned}(aP+b)^q(cP+d) - (aP+b)(cP+d)^q &= \prod_{(\alpha:\beta)\in\mathbb{P}^1(\mathbb{F}_q)} \beta(aP+b) - \alpha(cP+d) \qquad (E_m)\\ &= \prod_{(\alpha:\beta)\in\mathbb{P}^1(\mathbb{F}_q)} (\beta a - \alpha c)P + (\beta b - \alpha d)\\ &= \lambda \prod_{(\alpha:\beta)\in\mathbb{P}^1(\mathbb{F}_q)} P + \frac{\beta b - \alpha d}{\beta a - \alpha c}\\ &= \lambda \prod_{(\alpha:\beta)\in\mathbb{P}^1(\mathbb{F}_q)} P - m^{-1}\cdot(\alpha:\beta),\end{aligned}$$

where $P(X) - \infty$, if it appears, is by convention replaced by 1; indeed, when a/c is in $\mathbb{F}_q$, the expression $\beta a - \alpha c$ vanishes for a point $(\alpha:\beta) \in \mathbb{P}^1(\mathbb{F}_q)$ so that one of the factors of the product contains no term in $P(X)$.

The right-hand side is therefore, up to a multiplicative constant $\lambda \in \mathbb{F}_{q^2}^*$ a product of translates of the target $P(X)$ by elements of $\mathbb{F}_{q^2}$.

For the polynomial on the left-hand side of the equation, we need to reduce the degree of the systematic equation by using the special form of the defining polynomial: in K we have $x^q \equiv \frac{h_0(x)}{h_1(x)}$. Let us denote by a tilde the result of twisting elements of $\mathbb{F}_{q^2}$ by the Frobenius automorphism corresponding to q-th power computations. In particular, $\tilde{P}(X)$ is the polynomial $P(X)$ with all its coefficients raised to the power q. The left-hand side of the equation becomes

$$(\tilde{a}\tilde{P}(X^q) + \tilde{b})(cP(X)+d) - (aP(X)+b)(\tilde{c}\tilde{P}(X^q) + \tilde{d}),$$

and using the defining equation for the field K, it is congruent to

$$\mathcal{L}_m = \left(\tilde{a}\tilde{P}\left(\frac{h_0(X)}{h_1(X)}\right) + \tilde{b}\right)(cP(X)+d) - (aP(X)+b)\left(\tilde{c}\tilde{P}\left(\frac{h_0(X)}{h_1(X)}\right) + \tilde{d}\right).$$

As a rational fraction, this expression $\mathcal{L}_m$ has a denominator of degree less than δD and a numerator of degree less than $(1+\delta)D$. Since the denominator is a power of $h_1(X)$ and does not depend on m, from now on, when speaking about smooth elements, we assume that $h_1(X)$ is either indeed smooth, or has been added to the factor base elements. Hence, we are interested in the smoothness of the numerator of $\mathcal{L}_m$, and we say that $m \in \mathcal{P}_q$ yields a relation if this numerator of $\mathcal{L}_m$ is $\lceil D/2 \rceil$-smooth.

To any $m \in \mathcal{P}_q$, we associate a row vector $v(m)$ in the following way. Coordinates are indexed by $\lambda \in \mathbb{F}_{q^2}$, and the value associated to $\lambda \in \mathbb{F}_{q^2}$ is 1 or 0 depending on whether $P - \lambda$ appears in the right-hand side of the equation (E_m). Equivalently, we may write

$$v(m)_{\lambda \in \mathbb{F}_{q^2}} = \begin{cases} 1 \text{ if } (\lambda : 1) = \kappa m^{-1} \cdot (\alpha : \beta) \text{ with } (\alpha : \beta) \in \mathbb{P}^1(\mathbb{F}_q) \text{ and } \kappa \in \mathbb{F}_{q^2}^*, \\ 0 \text{ otherwise.} \end{cases}$$

We associate to the polynomial P a matrix whose rows are the vectors $v(m)$ for the cosets $m \in \mathcal{P}_q$ which yield a relation. The validity of Proposition 10.5.1 crucially relies on the following heuristic.

Heuristic 10.7.1. For any $P(X)$, the set of rows $v(m)$ for cosets $m \in \mathcal{P}_q$ that yield a relation form a matrix which has full rank.

If the matrix has full rank, then the vector $(\ldots, 0, 1, 0, \ldots)$ with 1 corresponding to $P(X)$ can be written as a linear combination of the rows. This linear combination expresses $\log P(X)$ as a linear combination logarithms of elements in the left-hand sides, that is polynomials of degree less than $\lceil D/2 \rceil$ as expected. Since there are $O(q^2)$ columns, the elimination process involves at most $O(q^2)$ rows, and since each row corresponds to an equation (E_m), it involves at most D polynomials in the left-hand-side[2]. In total, the polynomial D is expressed by a linear combination of at most $O(q^2 D)$ polynomials of degree less than $\lceil D/2 \rceil$. This linear algebra step costs $O(q^5)$ using sparse matrix techniques. □

10.8 Supporting the heuristic argument in the proof

Heuristic 10.7.1 affects the validity of the algorithm described in the previous section. We propose two approaches to support this heuristic. Both allow to gain some confidence in the validity of the heuristic, but of course none affect the heuristic nature of this statement.

A first line of justification calls for another heuristic argument, similar to heuristics which underpin the analysis of many sub-exponential factorization or discrete logarithm algorithms. Heuristic 10.7.1 considers the rank of a matrix which depends on P. We denote this matrix by $H(P)$. Using notations introduced in Section 10.7, we remark that the rows of $H(P)$ are a subset of the set of row vectors $v(m)$ for $m \in \mathcal{P}_q$. We denote by $\mathcal{H}$ the larger matrix whose rows are all these vectors $v(m)$. The matrix $\mathcal{H}$ has $\#\mathcal{P}_q = q^3 + q$ rows and q^2 columns.

If the numerator of $\mathcal{L}_m$ behaves like a random polynomial of similar degree (which we might heuristically expect), then the probability that it is smooth is constant. Indeed, the smoothness bound is a constant times smaller than the total degree. More precisely, we want a polynomial of degree $(1 + \delta)D$ to be

[2] This estimate of the number of irreducible factors is a pessimistic upper bound. In practice, one expects to have only $O(\log D)$ factors on average. Since the crude estimate does not change the overall complexity, we keep it that way to avoid adding another heuristic.

$\lceil D/2 \rceil$-smooth, which occurs with probability close to $\rho\big(2(1+\delta)\big)$. With δ being a constant, a constant proportion of the cosets m yield a relation, whence the matrix $H(P)$ is made of a constant fraction of the rows of the matrix $\mathcal{H}$. In other words, the matrix $H(P)$ has expected size $\Theta(q^3) \times q^2$, and Heuristic 10.7.1 is related to the probability of $H(P)$ having maximal rank q^2. Unless some theoretical obstruction occurs, we expect the probability to have a matrix that is not full rank to be in $O(1/q)$. The matrix $\mathcal{H}$ is however peculiar, and does enjoy regularity properties which are worth noticing. For instance, we have the following proposition.

Proposition 10.8.1. *Let ℓ be a prime not dividing $q^3 - q$. Then the matrix $\mathcal{H}$ over $\mathbb{F}_\ell$ has full rank q^2.*

Proof. If we extend $\mathcal{H}$ by one extra column corresponding to $\infty \in \mathbb{P}^1(\mathbb{F}_{q^2})$, thus forming a matrix $\mathcal{H}^+$, we have a bijection between rows of $\mathcal{H}^+$ and the different possible image sets of the projective line $\mathbb{P}^1(\mathbb{F}_q)$ within $\mathbb{P}^1(\mathbb{F}_{q^2})$, under injections of the form $(\alpha : \beta) \mapsto m^{-1} \cdot (\alpha : \beta)$. All these $q^3 + q$ image sets have size $q+1$, and by symmetry all points of $\mathbb{P}^1(\mathbb{F}_{q^2})$ are reached equally often. Therefore, the sum of all rows of $\mathcal{H}^+$ is the vector whose coordinates are all equal to $\frac{1}{1+q^2}(q^3+q)(q+1) = q^2 + q$. The same holds for the matrix $\mathcal{H}$.

Let us now consider the sum of the rows in $\mathcal{H}^+$ whose first coordinate is 1 (as we have just shown, we have $q^2 + q$ such rows). Those correspond to image sets of $\mathbb{P}^1(\mathbb{F}_q)$ which contain one particular point, say $(0 : 1)$. The value of the sum for any other coordinate indexed by e.g. $Q \in \mathbb{P}^1(\mathbb{F}_{q^2})$ is the number of image sets $m^{-1} \cdot \mathbb{P}^1(\mathbb{F}_q)$ which contain both $(0 : 1)$ and Q, which we prove is equal to $q + 1$ as follows. Without loss of generality, we may assume $Q = \infty = (1 : 0)$. We need to count the number of possible cosets m. We need to count the homographies $m^{-1} \in \mathrm{PGL}_2(\mathbb{F}_{q^2})$, modulo $\mathrm{PGL}_2(\mathbb{F}_q)$-equivalence $m \equiv hm$, fixing $(0 : 1)$ and $(1 : 0)$. Letting $m^{-1} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, we obtain $(b : d) = (0 : 1)$ and $(a : c) = (1 : 0)$, whence $b = c = 0$, and both $a, d \neq 0$. We may normalize to $d = 1$, and notice that multiplication of a by a scalar in $\mathbb{F}_q^*$ is absorbed in $\mathrm{PGL}_2(\mathbb{F}_q)$-equivalence. Therefore the number of suitable m is $\#\mathbb{F}_{q^2}^*/\mathbb{F}_q^* = q + 1$.

These two facts show that the row span of $\mathcal{H}^+$ (and, likewise, of $\mathcal{H}$) contains the vectors $(q^2+q, \ldots, q^2+q)$ and $(q^2+q, q+1, \ldots, q+1)$. The vector $(q^3-q, 0, \ldots, 0)$ is obtained as a linear combination of these two vectors, which suffices to prove that $\mathcal{H}^+$ and $\mathcal{H}$ have full rank, since the same reasoning holds for any coordinate. □

Proposition 10.8.1, while encouraging, is clearly not sufficient. We are, at the moment, unable to provide a proof of a more useful statement. On the experimental side, it is reasonably easy to sample arbitrary subsets of the rows of $\mathcal{H}$ and check for their rank. All trials we have made with random subsets of as few as q^2 rows have led to full rank matrices (experiments have been conducted up to $q = 2^5$).

A second line of justification is more direct and natural, as it is possible to implement the algorithm outlined in Section 10.7, and verify that it does provide the desired result. A Magma implementation validates this claim, and has been used to implement descent steps for an example field of degree 31 over $\mathbb{F}_{2^{10}}$. Of

course this tiny example size uses no optimization, and is only intended to check the validity of Proposition 10.5.1.

Finding appropriate h_0 and h_1. One key fact about the algorithm is the existence of two polynomials h_0 and h_1 in $\mathbb{F}_q[X]$ such that $h_1(X)X^q - h_0(X)$ has an irreducible factor of degree k. A partial solution is due to Joux [Jou13b] who showed how to construct such polynomials when $k \in \{q-1, q, q+1\}$. No such deterministic construction is known in the general case, but experiments show that one can apparently choose h_0 and h_1 of degree at most 2. We performed an experiment for every odd prime power q in $[3, 1000]$ and every $k \leq q$ and found that we could select $a \in \mathbb{F}_{q^2}$ such that $X^q + X^2 + a$ has an irreducible factor of degree k. Finally, note that the result joins a commonly made heuristic in discrete logarithm algorithms: for fixed $f \in \mathbb{F}_{q^2}[X, Y]$ and random $g \in \mathbb{F}_{q^2}[X, Y]$, the polynomial $\mathrm{Res}_Y(f, g)$ behaves as a random polynomial of same degree with respect to the degrees of its irreducible factors.

10.9 An improvement based on additional heuristics

The Heuristic 10.7.1 tells that a rectangular matrix with $\Theta(q)$ times more rows than columns has full rank. It seems reasonable to expect that only a constant times more rows than columns would be enough to get the full rank properties. Then, it means that we expect to have a lot of choices to select the best relations, in the sense that their left-hand sides split into irreducible factors of degrees as small as possible.

On average, we expect to be able to try $\Theta(q)$ relations for each row of the matrix. So, assuming that the numerators of $\mathcal{L}_m$ behave like random polynomials of similar degrees, we have to evaluate the expected smoothness that we can hope for after trying $\Theta(q)$ polynomials of degree $(1+\delta)D$ over $\mathbb{F}_{q^2}$. Set $u = \log q / \log\log q$, so that $u^u \approx q$. According to Theorem 1.3.1 it is then possible to replace $\lceil D/2 \rceil$ in Proposition 10.5.1 by the value $O(D \log\log q / \log q)$.

Then, the discussion leading to Theorem 10.5.2 can be changed to take this faster descent into account. We keep the same estimate for the arity of each node in the tree, but the depth is now only in $\log k / \log\log q$. Since this depth ends up in the exponent, the resulting complexity in Theorem 10.5.2 is then

$$q^{O(\log k / \log\log q)}.$$

10.10 Conclusion

The algorithm presented in this chapter achieves a dramatic improvement of the asymptotic complexity of discrete logarithm in finite fields, in almost the whole range of parameters where the Function Field Sieve was presently the most competitive algorithm. Compared to existing approaches, and in particular to the line of recent works [Jou13a, GGMZ13], the practical relevance of our algorithm is

not clear, and will be explored by further work. Various optimization and tuning strategies can be used in this context.

- Because of the arity of the descent tree, the breadth eventually exceeds the number of polynomials below some degree bound. It makes no sense, therefore, to use the descent procedure beyond this point, as the recovery of discrete logarithms of all these polynomials is better achieved as a pre-computation.

- The set of polynomials appearing in the right-hand side of Equation (E_m) in Section 10.7 is $\{P - \lambda\}$, because in the factorization of $Y^q - Y$, we substitute X with $m \cdot P$ for homographies m. In fact, we may apply m to $(P_0 : P_1)$ for two polynomials $P_{0,1}$ (P being one of them). This leads to factors of the form $\lambda_1 P_0 - \lambda_0 P_1$ in the right-hand sides. This variant is expected to lead to a practical improvement, and also offers opportunities for obtaining better smoothness estimates.

We note that the analysis of the algorithm presented here is heuristic, as discussed in Section 10.8. Some of the heuristics we stated, related to the properties of matrices $H(P)$ extracted from the matrix $\mathcal{H}$, seem accessible to more solid justification. It seems plausible to have the validity of algorithm rely on the sole heuristic of the validity of the smoothness estimates.

The crossing point between the $L(1/4)$ algorithm and our quasi-polynomial one is not determined yet. One of the key factors which hinders the practical efficiency of this algorithm is the $O(q^2 D)$ arity of the descent tree, compared to the $O(q)$ arity achieved by techniques based on Gröbner bases [Jou13a] at the expense of a $L(1/4 + \epsilon)$ complexity. Nevertheless, Adj et al. [AMORH13] proposed to mix the two algorithms and deduced that the new descent technique must be used for cryptographic sizes. Indeed, by estimating the time required to compute discrete logarithms in $\mathbb{F}_{3^{6 \cdot 509}}$, they showed the weakness of some pairing-based crypto-systems.

Conclusion and perspectives

In the present work we studied at length the discrete logarithm problem in finite fields. This classical topic in cryptography is less known than the closely related problem of factorization. We hence tried to offer a clear image of the family of discrete logarithm algorithms, with a special attention to their interactions.

First we focused on the notion of smoothness and on ECM, the fastest known smoothness test. We presented an improvement to the algorithm itself by analyzing the Galois properties of the division polynomials. We then used ECM to speed up the individual logarithm stage of NFS, reducing hence the asymptotic complexity.

Next we presented NFS and FFS and discussed some improvements. In particular we showed that, with the cost of a pre-computation depending on the bit-size of a range of primes p, one can reduce the complexity of discrete logarithms in all the fields $\mathbb{F}_p$ of that range.

Then we analyzed the polynomial selection stage of FFS. We showed how to measure the effect of cancellations and the advantage of inseparable polynomials. It makes possible to compare arbitrary polynomials with a unique function. This latter was sufficiently fast to allow us to select polynomials for two record computations.

Finally, we presented a new algorithm for fields of small characteristic. Following the recent improvements on the topic, we obtained a quasi-polynomial algorithm for finite fields of small characteristic. The key fact was to create a descent procedure which has a quasi-polynomial number of nodes, each requiring a polynomial time.

Perspectives

Galois group of torsion fields In Chapter **??** we found new sub-families of ECM-friendly families of curves. Can one implement an algorithm capable of listing all the sub-families with special properties when given a family of elliptic curves? The first step was to characterize these sub-families in terms of the Galois group. The next difficulty to overcome is to determine the equations which characterize each Galois group. A starting point can be the Resolvent Method [Coh93] of computing the Galois group. A second difficulty is to find a parametrization for the algebraic set described by the polynomial system hence obtained. For this we expect to have an effective solution with Gröbner bases since the examples seen in Chapter 3 make use of small systems of about 5 unknowns and one less equation. An application of such an algorithm would be to certify that a given family of

elliptic curves has no infinite sub-family of elliptic curves. A good milestone is Montgomery's conjecture that we discussed in Section 3.3.

The alpha function for NFS polynomial selection The complexity of the NFS algorithm relies on a series of conjectures. The most important one states that the norm of an algebraic number has the same smoothness probability as a random integer of the same size. The problem is actively researched in the community of analytic number theory [BBDT12]. Unfortunately, some of the ideas which stemmed when working with NFS and FFS remained confined to our community. For example Murphy's α, which proved its efficiency in practice, has not been studied from a theoretical point of view. This is why we launched a common work with Lachand aiming at first to study the properties of alpha.

The unit group conjecture in NFS A second conjecture in the NFS algorithm concerns the unit group and the Schirokauer maps. These maps allow to avoid any computations in the unit group but they increase the cost of the linear algebra stage in NFS by adding heavy columns to the matrix. A better understanding of the algorithms for the unit group could allow us to replace the Schirokauer maps with maps having smaller values. Indeed, the algorithms are sub-exponential with respect to the discriminant of the number field, in particular being very fast in SNFS. New ideas were introduced for the topic in [BF12].

Discrete logarithm with codes The new algorithm proposed in [AM12] has a complexity of type $L(1/2, \cdot)$, but it seems to be much faster than the other algorithms of the same type of complexity. In the factorization world the crossing point between the $L(1/2, \cdot)$ and $L(1/3, \cdot)$ algorithms is at the numbers of about 200 bits, so for the everyday computations in number theory the best choice are the $L(1/2)$ algorithms. We expect a similar situation for discrete logarithm algorithms. Hence it is interesting to improve Augot and Morain's algorithm.

Implementation of the recent DL algorithms Effectively using the new discrete logarithm algorithms can be a source of new ideas. Hence we plan to implement them first in high-level languages (Sage, Magma, ...), then in low level ones, particularly in C. The advantages of new algorithms are as follows.

- The relation collection stage seems to require less programming skills– FFS uses a memory of type $L(1/3)$.
- The new linear algebra stage consists in solving many linear systems rather than one of $L(1/3)$ unknowns as in the case of FFS. Hence one can solve reasonably large instances of the DLP using computer algebra programs. Additionally, since no virtual logarithms are involved, one can check the correctness of any equation independently on the others.

Improvements to the quasi-polynomial DL algorithm The new quasi-polynomial algorithm can be the object of numerous improvements.

- The algorithm relies on a series of assumptions. The most interesting assumption is that a matrix with $\Theta(q^3)$ random rows and $\Theta(q^2)$ columns has full rank.

- The setting of the algorithm requires to select polynomials h_1 and h_0 with coefficients in $\mathbb{F}_{q^2}$. If one selects these polynomials with coefficients in a subfield F of $\mathbb{F}_{q^2}$ then one can speed up the computations. Indeed, as in the Galois improvements of FFS and NFS, one can write, for each relation, one more relation for each $\sigma \in \mathrm{Gal}(\mathbb{F}_{q^2}/F)$.

- The Gröbner bases are new in the discrete logarithm alforithms. Hence the corresponding step of Joux's algorithm deserves a special attention.

Thinking on the long term After a long history of improvements over more than 30 years, the discrete logarithm remains a relatively difficult problem, but it continues to be a fertile domain of research.

We note first that discrete logarithm is still sub-exponential in fields of large characteristic. This confirms the predictions of Coppersmith [Cop84] who wrote: "In some sense [discrete logarithm modulo a prime] is to [factoring integers] what [discrete logarithms in $\mathrm{GF}(2^n)$] is to [factoring polynomials over $\mathrm{GF}(2)$]". Indeed, the best known algorithms for factorization and discrete logarithms in $\mathrm{GF}(p)$ are the two variants of NFS. Similarly, while the factorization in $\mathrm{GF}(2)[x]$ is probabilistic polynomial, discrete logarithms in $\mathrm{GF}(2^n)$ is quasi-polynomial.

Moreover, adapting the techniques of small characteristic to the large one is not enough to break the $L(1/3)$ barrier for discrete logarithms in $\mathrm{GF}(p)$ for primes p. Indeed, apart from the main phase which computes logarithms of the factor base, one also has to accelerate the individual logarithm stage. This latter stage has a complexity of $L(1/3)$ and, if one uses a smoothing step, then its complexity is independent on the cardinality of the factor base and of the algorithm which is used to complete the main phase. It is instead determined by the complexity of the smoothness tests, e.g. ECM. A faster factorization algorithm can determine us to use a factoring algorithm as smoothness test. Hence the fundamental problem of discrete logarithm remains a relatively difficult one, the most difficult case being the one of medium sized caracteristic.

Nevertheless, without breaking the $L(1/3)$ barrier, one might obtain improvements in the complexity of the various cases of the algorithm. It helps us learn more about objects like number and function fields as well as about the distribution of smooth numbers. Due to the fertility of the problem, with many improvements in a short time, discrete logarithm in finite fields is a less recommended primitive for cryptography than other problems whose presumably complexity is exponential: elliptic curves discrete logarithm, lattice based cryptography etc.

Algorithmes de logarithme discret dans les corps finis

Résumé

Introduction

Étant donnée sa difficulté estimée, le logarithme discret constitue avec la factorisation les deux problèmes les plus étudiés en cryptographie. De manière générale, dans un groupe quelconque, le logarithme discret de s en base t est le plus petit entier positif x tel que $t^x = s$. Sa difficulté dépend du groupe considéré, étant sous-exponentielle dans les groupes multiplicatifs des corps finis. Pour mesurer la complexité des problèmes sous-exponentiels, on introduit

$$L_n(\alpha, c) = \exp\left(c(\log n)^\alpha (\log\log n)^{1-\alpha}\right).$$

Notons que la complexité de la factorisation est $L_n(1/3, c)$, pour une constante positive c. Celle-ci était aussi la difficulté du logarithme discret dans les corps finis avant de récentes avancées. On énonce les différentes améliorations de la thèse en aboutissant sur un nouvel algorithme quasi-polynomial en petite caractéristique.

1 Probabilités de friabilité

L'ubiquité des résultats de friabilité nous a incités à dédier un chapitre à cet aspect. Un nombre entier est y-friable si tous ses diviseurs premiers sont plus petits que y en valeur absolue. On note $\Psi(x, y) = \{n \in [1, x], n \text{ est } y\text{-friable}\}$ et on dénote par $\psi(x, y)$ son cardinal.

Théorème 1 ([CEP83]). *Soit $\epsilon > 0$ fixé. Pour tout couple (x, u) de réels tels que $3 \leq u \leq (1-\epsilon)(\log x)/(\log\log x)$ on a :*

$$\psi(x, x^{1/u}) = x \exp\left\{-u\left(\log u + \log\log u - 1 + o(1)\right)\right\}.$$

Dans le cas où x et $x^{1/u}$ s'écrivent en fonction d'une autre quantité n à l'aide de la notation L_n, alors le théorème mentionné ci-dessus prend une forme simplifiée. De même, les calculs avec des quantités sous-exponentielles sont facilités par cette même notation.

Les innombrables similarités entre les entiers et les polynômes commencent avec leurs résultats de friabilité. Un polynôme est m-friable si tous ses facteurs irréductibles ont un degré inférieur à m.

Théorème 2 ([PGF98]). *Le nombre $N_q(n,m)$ des polynômes sur $\mathbb{F}_q$, m-friables et unitaires satisfait la relation suivante :*

$$N_q(n,m) = q^n \rho\left(\frac{n}{m}\right)\left(1 + O\left(\frac{\log n}{m}\right)\right),$$

où $O()$ est une fonction indépendante de q et ρ est la fonction de Dickman.

2 La méthode de factorisation par courbes elliptiques

Tester la friabilité d'un entier est un problème difficile pour lequel le meilleur algorithme est la méthode de factorisation par courbes elliptiques, abrégée ECM, et due à Lenstra [Len87]. Celle-ci repose sur la théorie des courbes elliptiques dont nous utilisons uniquement quelques résultats. Une courbe elliptique sous forme de Weierstrass sur un corps K est l'ensemble des solutions dans K^2 de l'équation $E : y^2 = x^3 + Ax + B$ dont on rajoute un point dit point à l'infini. Si p est un nombre premier, on appelle réduction de E modulo p la courbe elliptique sur $\mathbb{F}_p$ définie par $E(\mathbb{F}_p) : y^2 = x^3 + \overline{A}x + \overline{B}$, où la barre désigne la réduction modulaire. On admet qu'on peut définir une opération sur cet ensemble à l'aide de fonctions rationnelles et qu'on obtient ainsi un groupe commutatif. Si P est un point de la courbe et m un entier positif, on dénote par $[m]P$ la somme de P avec lui-même m fois.

L'algorithme consiste à répéter une sous-routine sur des paramètres choisis aléatoirement jusqu'à trouver un facteur strict. Si on arrête l'algorithme après un nombre donné de tests, on a une probabilité de 99% que le nombre n'est pas friable. Notons N le nombre à factoriser et p un facteur premier de N, inconnu au moment de l'exécution de l'algorithme. La sous-routine mentionnée plus haut commence par le choix d'une courbe E sur $\mathbb{Q}$ et d'un point $P = (x, y) \in E$ à coefficients dans $\mathbb{Q}$. L'étape suivante est de calculer un entier m qui est le produit de tous les nombres premiers jusqu'à une borne donnée, chacun élevé à une puissance de l'ordre de $\log N$. On calcule ensuite $Q = [m]P$. Si m est divisible par $\#E(\mathbb{F}_p)$ on vérifie que la coordonnée x de Q est divisible par p. Dans ce cas, on retrouve p en calculant $\mathrm{pgcd}(x, N)$. Pour calculer la probabilité de succès, Lenstra a montré que $\#E(\mathbb{F}_p)$ prend chaque valeur entière de l'intervalle $[p - \sqrt{p}, p + \sqrt{p}]$ avec une fréquence d'au moins $1/(2 \log p)$. La nature heuristique de l'algorithme est due exclusivement au manque de résultats suffisants sur la probabilité de friabilité d'un entier de l'intervalle précédent.

La littérature propose plusieurs axes d'amélioration pour ECM. Nous nous concentrons sur le choix de courbes adéquates, comme par exemple des courbes où l'arithmétique est efficace ou encore des courbes offrant de bonnes propriétés de friabilité. Montgomery [Mon92] a proposé les courbes $By^2 = x^3 + Ax + x$ qui permettent de calculer en utilisant uniquement la coordonnée x ; cela améliore l'arithmétique. Une nouvelle paramétrisation de courbes elliptiques a été proposée par Edwards [Edw07] et modifiée

ensuite dans [BL07, BBJ+08] pour obtenir les courbes d'Edwards tordues qui optimisent l'addition et le doublement. On note que toute courbe de Montgomery correspond à une courbe tordue d'Edwards et vice-versa. Quant à la probabilité de friabilité, Suyama et Montgomery ont proposé d'utiliser des courbes sur $\mathbb{Q}$ qui ont des points rationnels de torsion, ces derniers étant des points P qui ont un multiple $[m]P$ égal à l'élément neutre. Le théorème de Mazur limite le nombre de possibilités pour le choix des points de torsion. La recherche des courbes correspondantes a été exhaustive. Nous proposons des familles de courbes qui ont une meilleure probabilité de succès sans avoir une meilleure torsion sur $\mathbb{Q}$.

3 Familles de courbes pour ECM

Afin de caractériser les courbes elliptiques qui accélèrent l'algorithme ECM, on commence par prouver un résultat théorique. Pour toute courbe elliptique E et tout entier positif m on note $E[m]$ l'ensemble des points de m-torsion et par $\mathbb{Q}(E[m])$ le plus petit corps de nombres qui contient les coordonnées des points de $E[m]$. Soit f le polynôme de définition de $K = \mathbb{Q}(E[m])$, θ une racine de f dans $\overline{\mathbb{Q}}$ et p un premier quelconque. Pour tout automorphisme $\sigma \in \mathrm{Gal}(K)$, déterminé par $\theta \mapsto A(\theta)$ avec $A \in \mathbb{Z}[x]$, on définit sa réduction modulo p par $\overline{\theta} \mapsto A(\overline{\theta})$ où $\overline{\theta}$ est une racine de f dans $\overline{\mathbb{F}_p}$. On appelle Frobenius de $\mathbb{Q}(\theta)$ modulo p le sous-ensemble de $\mathrm{Gal}(K)$ dont la réduction modulo p correspond à l'automorphisme $x \mapsto x^p$. Le théorème de Chebotarev affirme que, pour tout corps galoisien, la densité naturelle de premiers p tels que le Frobenius de K modulo p correspond à une classe de conjugaison C de $\mathrm{Gal}(K)$ est égale à $\#C/\#\,\mathrm{Gal}(K)$.

Le corps $\mathbb{Q}(E[m])$ peut être calculé effectivement à l'aide des polynômes dits de m-division. Ses automorphismes se représentent de manière unique comme matrices de $\mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z})$. Le point de départ de notre résultat est l'observation qu'une courbe E sur $\mathbb{Q}$ a des points de m-torsion modulo un premier p si et seulement si le Frobenius de $\overline{\mathbb{F}_p}$ fixe des points de $E(\mathbb{F}_p)[m]$ autres que l'élément neutre. On remarque ensuite que tout automorphisme de K a la même matrice dans $\mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z})$ que sa réduction modulo p. Quand m est une puissance de premier π^k, on relie alors la proportion de premiers tels que m divise $\#E(\mathbb{F}_p)$ à la proportion de matrices dans l'image de $\mathrm{Gal}(K)$ dans $\mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z})$ qui fixent au moins un point non-nul. En particulier, pour $m = \pi$ on a le résultat suivant.

Corollaire 3. *Soit E une courbe elliptique et π un nombre premier. Alors on a :*

$$\mathrm{Prob}\left(E(\mathbb{F}_p)[\pi] \simeq \mathbb{Z}/\pi\mathbb{Z}\right) = \frac{\#\{g \in \rho_\pi(\mathrm{Gal}(\mathbb{Q}(E[\pi])/\mathbb{Q})) \mid \det(g - \mathrm{Id}) = 0, g \neq \mathrm{Id}\}}{\#\,\mathrm{Gal}(\mathbb{Q}(E[\pi])/\mathbb{Q})},$$

$$\mathrm{Prob}\left(E(\mathbb{F}_p)[\pi] \simeq \mathbb{Z}/\pi\mathbb{Z} \times \mathbb{Z}/\pi\mathbb{Z}\right) = \frac{1}{\#\,\mathrm{Gal}(\mathbb{Q}(E[\pi])/\mathbb{Q})}.$$

Dans la suite on considère que E est sans multiplication complexe. Un théorème de Serre [Ser71] caractérise les images de $\mathrm{Gal}(\mathbb{Q}(E[\pi^k]))$ dans $\mathrm{GL}_2(\mathbb{Z}/\pi^k\mathbb{Z})$ quand π est premier et $k \geq 1$. En effet, l'application est surjective à l'exception d'un nombre fini de premiers π. De plus, pour tout premier π et pour k suffisamment grand, chaque matrice de l'image de $\mathrm{Gal}(\mathbb{Q}(E[\pi^k]))$ se relève en exactement π^4 matrices de $\mathrm{Gal}(\mathbb{Q}(E[\pi^{k+1}]))$.

D'après un troisième résultat de Serre [Ser81], pour montrer la surjectivité de l'injection pour toute paire (π, k), il suffit de tester la surjectivité pour les paires $(\pi = 2, k = 3)$, $(\pi = 3, k = 2)$ et $(\pi, k = 1)$ quand $\pi \geq 5$. Si on applique les théorèmes de Serre et on utilise notre caractérisation, on obtient un résultat bien connu en pratique. Celui-ci affirme que, si p est un premier aléatoire, la probabilité que $\#E(\mathbb{F}_p)$ est divisible par une puissance de premier π^k est égale à celle d'un entier quelconque, à un facteur constant près, indépendant de k. C'est ce facteur constant qui fait la différence entre les différentes courbes elliptiques.

Armés des outils théoriques, nous pouvons maintenant caractériser les sous-familles des paramétrisations utilisées en pratique. Ainsi, nous nous sommes intéressés dans [BBB+12] au courbes d'Edwards d'équation $-x^2+y^2 = 1-e^4x^2y^2$, qui ont une torsion $\mathbb{Z}/2\mathbb{Z}\times\mathbb{Z}/2\mathbb{Z}$ sur $\mathbb{Q}$. Afin de modifier leur groupe de Galois, on a choisit de diminuer le degré de $\mathbb{Q}(E[4])$ en choisissant des valeurs de e qui factorisent un ou plusieurs facteurs du polynôme de 4-division. Ainsi on a trouvé les équations $e = g^2$, $e = (2g^2 + 2g + 1)/(2g + 1)$, $e = g^2/2$ et $e = (q - 1/g)/2$. On remarque que cela explique toutes les courbes qui avaient été remarquées par Bernstein et al. [BBLP13].

On englobe ensuite un résultat antérieur de l'auteur qui concerne les courbes de Suyama ayant de bonnes propriétés de friabilité. Ainsi les paramètres $\sigma = 11$ et $\sigma = 9/4$ ont des groupes de Galois différents des autres courbes de Suyama. Une analyse des points de 4-torsion et de 8-torsion nous suggère des équations à imposer. Les résultats théoriques nous permettent de calculer l'amélioration d'efficacité. Ces résultats confirment les observations de Bernstein et al. sur la traduction de la famille de Suyama en coordonnées d'Edwards. Leur étude avait révélée la même efficacité pour cette famille avec 6 points de torsion que pour la famille de 12 points de torsion sur $\mathbb{Q}$. Cela se rajoute à une meilleure arithmétique pour la famille à 6 points de torsion.

4 Améliorations du problème de friabilisation

La première partie de l'étape de logarithme individuel des algorithmes de logarithme discret peut être modélisée comme un problème d'algorithmique, qu'on appelle friabilisation. Comme données d'entrée, on a une liste infinie d'entiers appartenant à $[1, n]$, ou de manière équivalente un générateur de nombres aléatoires. On peut également utiliser une boîte noire qui teste la friabilité ayant la même complexité que ECM et qui renvoie la partie friable. Il est demandé de trouver dans le moindre temps un entier de la liste qui est $L_n(\alpha, c)$-friable pour les paramètres $\alpha \in]0, 1[$ et $c > 0$ de notre choix.

La méthode naïve consiste à considérer les entiers à la suite et à faire un test de friabilité. Nous proposons un algorithme en deux, puis en plusieurs, étapes. La première étape utilise le test de friabilité avec une petite borne de friabilité. Les candidats qui réussissent le premier test sont appelés admissibles et ils sont soumis à un deuxième test avec une borne de friabilité plus grande. La raison du test en deux étapes est que, pour toute constante θ de $]0, 1[$, il existe $c > 0$ tel qu'une grande proportion des nombres B-friables n a une partie B^θ-friable au moins égale à n^c.

Théorème 4. *Soit a un réel positif. Pour tout entier n, on pose $L(n) = L_n(2/3, a)$, abrégé aussi L. Soient c et θ deux constantes positives dans l'intervalle $]0, 1[$. Notons M_1 l'ensemble des entiers de $[1, n]$ dont la partie L^θ-friable est plus grande que n^c. De même, notons M_2 le sous-ensemble des éléments L-friables de M_1. On a alors :*

(i) $\#M_1 \leq nL_n\left(1/3, \frac{c}{3a\theta}\right)^{-1+o(1)}$;

(ii) $\#M_2 \geq nL_n\left(1/3, \frac{c}{3\theta a} + \frac{1-c}{3a}\right)^{-1+o(1)}$.

Après l'optimisation des paramètres c et θ, on trouve un algorithme de complexité $L_p(1/3, 1.296)$, améliorant ainsi l'algorithme naïf dont la complexité est $L_p(1/3, 1.442)$. À l'aide de plusieurs tests d'admissibilité on obtient une complexité de $L_p(1/3, 1.232)$.

5 Algorithmes élémentaires de logarithme discret

Avant de présenter des algorithmes rapides qui font intervenir des outils mathématiques relativement avancés, on commence par des algorithmes généraux. Ceux-ci continuent d'être utilisés en conjonction avec les nouveaux algorithmes afin d'écarter une série de difficultés techniques. En effet, l'algorithme de Pohlig-Hellman réduit le problème au calcul du logarithme discret modulo un facteur premier ℓ du cardinal du groupe. L'algorithme ρ de Pollard rend possible le calcul des logarithmes modulo ℓ dans $O(\sqrt{\ell})$ opérations, permettant ainsi de supposer que ℓ est très grand.

Passons maintenant aux algorithmes spécifiques aux corps finis. Premier algorithme de sa famille, Index calculus permet de calculer des logarithmes discrets dans les corps $\mathbb{Z}/p\mathbb{Z}$ avec p premier. Soient t et s deux éléments du groupe et supposons que t est un générateur. On commence par calculer des relations $(t^h \bmod p) = \prod p_i$ où les nombres premiers p_i qui apparaissent au membre droit sont inférieurs à un entier B dit borne de friabilité. Chaque relation permet d'écrire une équation entre les logarithmes discrets en base t : $h = \sum e_i \log_t p_i$ où e_i est la valuation de p_i dans $(t^h \bmod p)$. Une étape d'algèbre linéaire résout le système formé par ces équations et trouve $\log_t p_i$ pour tout premier p_i apparaissant dans le système. Finalement, une étape appelée logarithme individuel relie $\log_t s$ aux logarithmes des premiers p_i.

L'algorithme s'étend aux corps à 2^n éléments sans aucune difficulté. Toujours sans utiliser des outils sophistiqués, on présente un algorithme de complexité $L(1/3)$. N'étant pas le premier algorithme de cette complexité, cet algorithme proposé par Joux et Lercier a l'avantage de la simplicité.

6 Présentation des algorithmes NFS et FFS

Après les travaux de Joux, Lercier, Smart et Vercauteren [JLSV06], on peut calculer les logarithmes discrets dans tous corps finis $\mathbb{F}_{p^n}$ dans un temps $L_{p^n}(1/3, c)$ avec $c > 0$. Selon la taille relative de p et n, on utilise le crible algébrique (NFS), le crible des corps de fonctions (FFS) ou le crible algébrique de haut degré (NFS-HD), conformément à la Figure 1. Le crible algébrique pour logarithme discret suit les mêmes lignes que Index

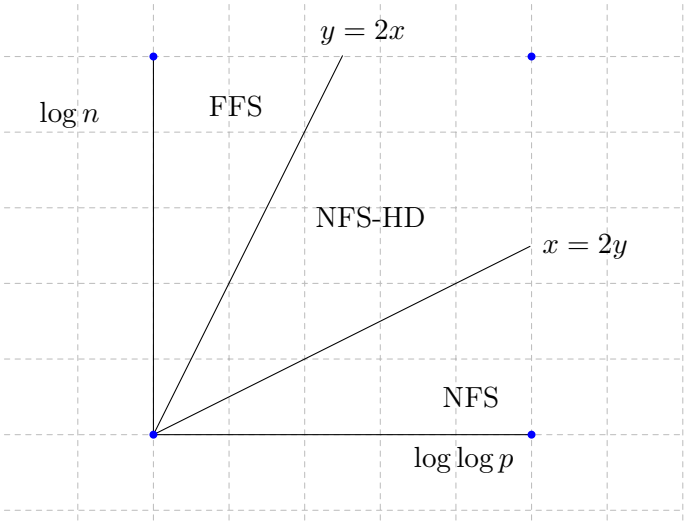


FIGURE 1: Le domaine d'application de FFS, NFS-HD et NFS selon $\mathbb{F}_Q^*$ avec $Q = p^n$.

calculus. On commence par choisir deux polynômes f et g dans $\mathbb{Z}[x]$ irréductibles qui partagent une racine m modulo p. Les relations sont des polynômes $a - bx \in \mathbb{Z}[x]$ tels que $N_f = b^{\deg f} f(a/b)$ et $N_g = b^{\deg g} g(a/b)$ sont friables. Les deux quantités N_f et N_g portent une notion de taille qui caractérise les éléments $a - b\alpha_f$ et $a - b\alpha_g$ où α_f et α_g sont des racines algébriques de f et g. Ces nombres se factorisent en idéaux de corps de nombres, par exemple $a - b\alpha_f = \prod \mathfrak{p}_i$. À condition de résoudre une série de difficultés techniques, on peut écrire des équations additives comme suit :

$$\sum_i e_i(a,b) \log \mathfrak{p}_i + \sum_j e'_j \lambda_j(a,b) \equiv \sum_i f_i(a,b) \log \mathfrak{q}_i + \sum_j f'_j \lambda'_j(a,b) \mod \ell, \quad (1)$$

où $\mathfrak{p}_i$ et $\mathfrak{q}_i$ sont des idéaux premiers des corps de nombres de f et g et où il reste à définir les logarithmes et les fonctions λ qui interviennent. Pour cela on définit les fonctions de Schirokauer, notées λ_j et λ'_j, apparentées au logarithme p-adique. Les logarithmes des idéaux sont appelés logarithmes virtuels et ils sont définis comme suit. Soit h le nombre de classes du corps défini par f. Le logarithme de $\mathfrak{p}$, noté $\log \mathfrak{p}$, est égal au logarithme discret de z, le générateur de $\mathfrak{p}^h$ qui a une valeur nulle sur toutes les fonctions de Schirokauer.

Le crible des corps de fonctions est similaire à NFS. La principale différence vient du fait qu'un corps fini $\mathbb{F}_{p^n}$ peut être représenté par de multiples polynômes irréductibles $\varphi \in \mathbb{F}_p[x]$ de degré n. Ainsi, on peut commencer par choisir un polynôme f de bonne qualité pour lequel on choisit un polynôme aléatoire g tel que $\mathrm{Res}(f,g)$ a un facteur irréductible de degré n. De façon heuristique, cela arrive avec probabilité $1/n$ et permet de choisir φ égal au facteur mentionné plus haut. Les autres avantages de FFS par rapport à NFS sont le test de friabilité qui est polynomial, à la place du test sous-exponentiel (ECM) pour NFS, et le remplacement des fonctions de Schirokauer par les valuations aux places à l'infini.

7 Complexités pour NFS et FFS

Afin d'avoir une présentation complète du sujet, on consacre quelques pages à la complexité des principaux algorithmes. Cela nous permet en particulier d'étudier une version de NFS appelée l'usine de logarithmes discrets. Inspirés d'un algorithme de factorisation, on propose d'utiliser un polynôme $g = x - m$ commun pour tous les premiers p d'une taille donnée. Ainsi on collecte les paires (a, b) telles que $a - bm$ est friable et on les stocke dans un fichier persistant. Pour chaque premier p, on parcourt le fichier précédemment constitué et on teste la friabilité de $b^{\deg f} f(a/b)$ à l'aide d'ECM. Cela donne un précalcul de complexité $L_p(1/3, 2.007)$ pour constituer le fichier mais une complexité de $L_p(1/3, 1.638)$ pour tout calcul ultérieur. L'obstacle pratique est la taille prohibitive du fichier, qui vaut $L_p(1/3, 1.638)$.

8 Améliorations à NFS et FFS

L'algorithme NFS n'est jamais implémenté sans deux améliorations qui ont été proposées ultérieurement et que nous avons préféré omettre dans la première description de l'algorithme. Il s'agit de la version par blocs de l'algèbre linéaire, qui permet de paralléliser les calculs, et de la version par spécial-Q pour l'étape de collecte de relations. À cela, on rajoute une amélioration personnelle et on explique l'efficacité d'une modification de [JLSV06]. Plus en détail, on propose une nouvelle sélection polynomiale de NFS pour les corps $\mathbb{F}_{p^n}$ quand $n > 1$ en généralisant une méthode de Joux et Lercier ; elle était utilisée précédemment uniquement pour les corps premiers.

9 Sélection polynomiale pour FFS

Les calculs effectifs de logarithmes discrets montrent qu'il existe des différences importantes d'efficacité pour FFS en fonction des polynômes f utilisés. Un bon choix permet de gagner un facteur 2 dans l'étape de collecte de relations. L'idée que nous avons utilisé dans [Bar13] est de définir des fonctions facilement calculables qui déterminent l'efficacité d'un polynôme, à l'instar des méthodes utilisées pour la factorisation [Mur99]. Ces fonctions sont ensuite calculées pour tous les polynômes qui minimisent ce qu'on appelle la propriété de taille, notée σ, ou de manière équivalente le degré moyen de la quantité N_f vue plus haut.

La première propriété des polynômes adaptés à FFS est le fait d'avoir un grand nombre de racines modulo des petits polynômes irréductibles. Par racine de $f(t, x)$ modulo $\ell(t) \in \mathbb{F}_q[t]$ on entend un polynôme $r(t)$ de degré plus petit que $\ell(t)$ tel que $f(t, r(t))$ est divisible par $\ell(t)$. L'analyse détaillée dans la thèse montre que la probabilité que N_f soit divisible par $\ell(t)$ est essentiellement proportionnelle au nombre de racines de f modulo $\ell(t)$. Ainsi, un polynôme adapté a une bonne probabilité que N_f ait des petits facteurs et par conséquent une meilleure probabilité de friabilité. Pour quantifier cette propriété, dite propriété des racines modulaires, on définit la quantité $\alpha_\ell(f)$. Celle-ci mesure la différence de degré entre la partie $\ell(t)$-friable d'un polynôme quelconque et, respectivement,

de la quantité N_f. Finalement, on définit la série $\alpha(f) = \sum_\ell \alpha_\ell(f)$ dont la convergence est basée sur le théorème de Hasse-Weil.

La deuxième propriété étudiée est celle des annulations des termes. Cela concerne les paires $(a, b) \in \mathbb{F}_q[t]^2$ pour lesquelles le degré de N_f n'est pas égal à sa valeur attendue : $\deg_t f + \deg_x f \max(\deg a, \deg b)$. On caractérise les paires avec cette propriété à l'aide des polynômes de Laurent r tels que $f(r)$ a un degré inférieur à $\deg_t f + \deg_x f \deg r$, où le degré d'une fraction rationnelle est la différence des degrés de son numérateur et son dénominateur. Pour mesurer cette propriété, on introduit la fonction α_∞ qui correspond au degré moyen gagné grâce aux annulations.

En combinant les fonctions précédentes, on souhaite comparer des polynômes aléatoires. On définit ϵ par $\sigma + \alpha + \alpha_\infty$, qui donne l'intuition d'un degré équivalent pour les polynômes aléatoires qui ont la même probabilité de friabilité que N_f. La correction de ϵ n'est pas prouvée, mais on illustre sa fiabilité de manière expérimentale. Nos résultats indiquent qu'un polynôme qui a une meilleure valeur de ϵ peut être au plus 5% moins efficace. Ainsi ϵ est une alternative plus rapide pour la fonction $\mathbb{E}$, qu'on peut définir par analogie avec les cas de la factorisation.

L'utilisation des polynômes inséparables ($\frac{\partial f}{\partial x} = 0$) et de l'algorithme de Coppersmith dans les records n'est pas un accident. En effet, les mauvaises valeurs de α sont compensées pour ces polynômes par un très grand nombre de relations dites gratuites, qui sont générées dans un temps négligeable. Cela permet de voir l'algorithme de Coppersmith comme une cas particulier de FFS et de ne pas distinguer entre les deux algorithmes lors des améliorations pratiques.

Ayant les outils théoriques dans notre possession, il a été possible de tester un nombre de 2^{48} polynômes f, dont on a gardé les dix meilleurs pour un test direct. Les plus efficaces ont été utilisés dans les records de calcul de logarithme discret par l'équipe Caramel, correspondant à $\mathbb{F}_{2^{619}}$ et $\mathbb{F}_{2^{809}}$. On revoit aussi d'autres records de logarithmes discrets et on cherche des alternatives aux polynômes utilisés.

10 Un algorithme quasi-polynomial

Suite aux avancées récentes de Joux [Jou13], le cas des corps finis de très petite caractéristique s'est avéré avoir une complexité de type $L(1/4)$. Dans [BGJT13], nous proposons une amélioration qui calcule le logarithme dans $\mathbb{F}_Q$ en temps quasi-polynomial $(\log Q)^{O(\log \log Q)}$.

Un point important est le choix des polynômes φ qui représentent $\mathbb{F}_Q$ pour $Q = q^{2k}$. On choisit de manière aléatoire des polynômes h_0 et h_1 de degré 2 dans $\mathbb{F}_{q^2}[x]$ jusqu'à trouver une paire telle que $h_1 x^q - h_0$ a un diviseur irréductible φ de degré k. Cela arrive de manière heuristique avec une probabilité de $1/k$. On note que, si x est une racine de φ dans $\mathbb{F}_Q$, alors $x^q = \frac{h_0(x)}{h_1(x)}$. La brique de base se résume dans le résultat suivant.

Proposition 5. *Soit $K = \mathbb{F}_{q^{2k}}$ un corps fini pour lequel il existe des polynômes h_0 et h_1 comme ci-dessus. Sous des heuristiques bien spécifiées, il existe un algorithme de complexité polynomiale en q et k qui peut effectuer les deux tâches suivantes.*

1. *Étant donné un élément de K représenté par un polynôme $P \in \mathbb{F}_{q^2}[X]$ avec $2 \leq \deg P \leq k-1$, l'algorithme fournit une expression de $\log P(x)$ comme combinaison linéaire d'au plus $O(kq^2)$ logarithmes $\log P_i(x)$ avec $\deg P_i \leq \lceil \frac{1}{2} \deg P \rceil$ et de $\log h_1(x)$.*
2. *L'algorithme fournit le logarithme de $h_1(x)$ et ceux des éléments de K de la forme $x+a$, pour tout a dans $\mathbb{F}_{q^2}$.*

L'algorithme ci-dessus permet de construire un algorithme complet de logarithme discret. En effet, on exprime le logarithme du polynôme donné en fonction de polynômes de degré $(\deg P)/2$, ensuite $(\deg P)/2^2$ et ainsi de suite jusqu'au polynômes de degré 1. Comme chaque étape prend un temps polynomial en q et k, ayant tous les deux une taille de l'ordre de $\log Q$, il suffit de compter le nombre de noeuds de l'arbre ainsi obtenu. La hauteur de l'arbre étant $\log_2 k$ et son arité $O(kq^2)$, on trouve un nombre $q^{O(\log k)} = (\log Q)^{O(\log \log Q)}$.

Conclusion

Après avoir vu comment améliorer l'algorithme ECM de factorisation, on montre comment l'intégrer dans un algorithme de logarithme discret. La présentation générale des algorithmes phares de logarithme discret, le crible algébrique et le crible des corps des fonctions, nous a permis de proposer des améliorations. On a ainsi vu une nouvelle version de l'algorithme, basée sur le stockage d'un fichier persistant, et on a proposé une modification pour NFS dans le cas non-premier. On a vu ensuite comment accélérer FFS par le choix de polynômes adéquats. Enfin, on a proposé un algorithme qui est asymptotiquement meilleur que FFS. Parmi les problèmes qui restent ouverts, on remarque l'utilisation de notre caractérisation des courbes elliptiques adaptées à ECM afin de trouver des nouvelles familles par une méthode systématique. De même, le nouvel algorithme quasi-polynomial peut faire l'objet de nombreuses améliorations.

Bibliography

[AD93] L. M. Adleman and J. DeMarrais. A subexponential algorithm for discrete logarithms over all finite fields. *Math. Comp.*, 61(203):1–15, 1993.

[Adl79] L. M. Adleman. A subexponential algorithm for the discrete logarithm problem with applications to cryptography. In *20th Annual Symposium on Foundations of Computer Science*, pages 55–60. IEEE, 1979.

[Adl94] L. M. Adleman. The function field sieve. In *Algorithmic Number Theory–ANTS I*, volume 877 of *Lecture Notes in Comput. Sci.*, pages 108–121. Springer, 1994.

[AH99] L. M. Adleman and M. D. A. Huang. Function field sieve method for discrete logarithms over finite fields. *Information and Computation*, 151(1):5–16, 1999.

[Alb11] M. R. Albrecht. The M4RI & M4RIE libraries for linear algebra over $\mathbb{F}_2$ and small extensions, 2011. Talk at the Sage Days 35, Warwick, UK.

[Alb12] M. R. Albrecht. The M4RIE library for dense linear algebra over small fields with even characteristic. In *International Symposium on Symbolic and Algebraic Computation–ISSAC '12*, pages 28–34. ACM, 2012.

[AM93] A. O. L. Atkin and F. Morain. Finding suitable curves for the elliptic curve method of factorization. *Math. Comp.*, 60(201):399–405, 1993.

[AM12] D. Augot and F. Morain. Discrete logarithm computations over finite fields using reed-solomon codes, 2012. Preprint, available on arXiv:1202.4361.

[AMORH13] G. Adj, A. Menezes, T. Oliveira, and F. Rodríguez-Henríquez. Weakness of $\mathbb{F}_{3^{6\cdot 509}}$ for discrete logarithm cryptography, 2013. Cryptology ePrint Archive Report 2013/446.

[Apo90] T. M. Apostol. *Modular functions and Dirichlet series in number theory*, volume 41 of *Graduate Texts in Mathematics*. Springer-Verlag, 1990.

[Bai11] S. Bai. *Polynomial selection for the number field sieve*. PhD thesis, Australian National Univers., 2011.

[Bar13] R. Barbulescu. Selecting polynomials for the function field sieve, 2013. Cryptology ePrint Archive Report 2013/200, submitted in March 2013.

[BB07] A. T. Benjamin and C. D. Bennett. The probability of relatively prime polynomials. *Mathematics Magazine*, 80(3):196–202, 2007.

[BBB+12] R. Barbulescu, J. W Bos, C. Bouvier, T. Kleinjung, and Peter L. Montgomery. Finding ECM-friendly curves through a study of Galois properties. In *Algorithmic Number Theory–ANTS X*, 2012. To appear.

[BBD+12] R. Barbulescu, C. Bouvier, J. Detrey, P. Gaudry, H. Jeljeli, E. Thomé, M. Videau, and P. Zimmermann. The relationship between some guy and cryptography, 2012. rump session of ECC, humoresque.

[BBD+13] R. Barbulescu, C. Bouvier, J. Detrey, P. Gaudry, H. Jeljeli, E. Thomé, M. Videau, and P. Zimmermann. Discrete logarithm in GF(2^{809}) with FFS, 2013. Cryptology ePrint Archive Report 2013/197.

[BBDT12] A. Balog, V. Blomer, C. Dartyge, and G. Tenenbaum. Friable values of binary forms. *Commentarii Mathematici Helvetici*, 87(3):639–667, 2012.

[BBJ+08] D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters. Twisted Edwards curves. In *Progr. in Cryptology–AFRICACRYPT 2008*, volume 5023 of *Lecture Notes in Comput. Sci.*, pages 389–405. Springer, 2008.

[BBL10] D. J. Bernstein, P. Birkner, and T. Lange. Starfish on strike. In *Progr. in Cryptology–LATINCRYPT 2010*, volume 6212 of *Lecture Notes in Comput. Sci.*, pages 61–80. Springer, Heidelberg, 2010.

[BBLP13] D. J. Bernstein, P. Birkner, T. Lange, and C. Peters. ECM using Edwards curves. *Math. Comp.*, 82(282):1139–1179, 2013.

[BC10] E. Brier and C. Clavier. New families of ECM curves for Cunningham numbers. In *Algorithmic Number Theory–ANTS IX*, volume 6197 of *Lecture Notes in Comput. Sci.*, pages 96–109. Springer, 2010.

[BCP97] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[BDEZ12] R. Barbulescu, J. Detrey, N. Estibals, and P. Zimmermann. Finding optimal formulae for bilinear maps. In *Arithmetic of Finite Fields–WAIFI 2012*, volume 7369 of *Lecture Notes in Comput. Sci.*, pages 168–186. Springer, 2012.

[Ber04] D. J. Bernstein. How to find smooth parts of integers, 2004. Preprint available at http://cr.yp.to/.

[BF12] J.-F. Biasse and C. Fieker. Improved techniques for computing the ideal class group and a system of fundamental units in number fields. In *Algorithmic Number Theory–ANTS X*, 2012. To appear.

[BFG+09] S. Bai, A. Filbois, P. Gaudry, A. Kruppa, F. Morain, E Thomé, P. Zimmermann, et al. Crible algébrique: Distribution, optimisation – NFS, 2009. Downloadable at http://cado-nfs.gforge.inria.fr/.

[BFHMV84] I. Blake, R. Fuji-Hara, C. Mulliní, and SA Vanstone. Computing logarithms in finite fields of characteristic two. *Computing*, 5(2), 1984.

[BGJT13] R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic, 2013. Cryptology ePrint Archive Report 2013/400.

[BL07] D. J. Bernstein and T. Lange. Faster addition and doubling on elliptic curves. In *Advances in Cryptology–ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Comput. Sci.*, pages 29–50. Springer, 2007.

[BL11] D. J. Bernstein and T. Lange. A complete set of addition laws for incomplete Edwards curves. *J. Number Theory*, 131(5):858–872, 2011.

[BL12] D. J. Bernstein and T. Lange. Two grumpy giants and a baby. In *Algorithmic Number Theory–ANTS X*, 2012. To appear.

[Bou13] C. Bouvier. The filtering step of discrete logarithm and integer factorization algorithms, 2013. Preprint available at http://hal.inria.fr/hal-00734654.

[BSS99] I.F. Blake, G. Seroussi, and N.P. Smart. *Elliptic curves in cryptography*, volume 265. Cambridge University Press, 1999.

[Cav00] S. Cavallar. *Strategies in filtering in the number field sieve.* PhD thesis, Univers. Leinden, 2000.

[CBH11] N. T. Courtois, G. V. Bard, and D. Hulme. A new general-purpose method to multiply 3×3 matrices using only 23 multiplications, 2011. Preprint, available on arXiv:1108.2830.

[CEP83] E. R. Canfield, P. Erdös, and C. Pomerance. On a problem of Oppenheim concerning "factorisatio numerorum". *Journal of Number Theory*, 17(1):1–28, 1983.

[CH07] J. Chung and M. A. Hasan. Asymmetric squaring formulae. In *IEEE Symposium on Computer Arithmetic–ARITH 18*, pages 113–122. IEEE, 2007.

[CKO09] M. Cenk, C.K. Koç, and F. Ozbudak. Polynomial multiplication over finite fields using field extensions and interpolation. In *IEEE Symposium on Computer Arithmetic–ARITH 19*, pages 84–91. IEEE, 2009.

[CÖ08] M. Cenk and F. Özbudak. Efficient multiplication in $\mathbb{F}_{3^{\ell m}}$, $m \geq 1$ and $5 \leq \ell \leq 18$. In *Progr. in Cryptology–AFRIKACRYPT 2008*, volume 5023 of *Lecture Notes in Comput. Sci.*, pages 406–414, 2008.

[CO09] M. Cenk and F. Ozbudak. Improved polynomial multiplication formulas over $\mathbb{F}_2$ using Chinese remainder theorem. *IEEE Trans. Comput.*, 58(4):572–576, 2009.

[CÖ10] M. Cenk and F. Özbudak. On multiplication in finite fields. *J. Complexity*, 26:172–186, 2010.

[CÖ11] M. Cenk and F. Özbudak. Multiplication of polynomials modulo x^n. *Theoret. Comput. Sci.*, 412:3451–3462, 2011.

[Coh93] H. Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, 1993.

[Cop84] D. Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Trans. Inform. Theory*, 30(4):587–594, 1984.

[Cop94] D. Coppersmith. Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. *Math. Comp.*, 62(205):333–350, 1994.

[COS86] D. Coppersmith, A. M. Odlyzko, and R. Schroeppel. Discrete logarithms in GF(p). *Algorithmica*, 1(1-4):1–15, 1986.

[CS06] A. Commeine and I. Semaev. An algorithm to solve the discrete logarithm problem with the number field sieve. In *Public Key Cryptology–PKC 2006*, volume 3958 of *Lecture Notes in Comput. Sci.*, pages 174–190. Springer, 2006.

[DGV13] J. Detrey, P. Gaudry, and M. Videau. Relation collection for the function field sieve. In *IEEE Symposium on Computer Arithmetic–ARITH 21*. IEEE, 2013.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, 22(6):644–654, 1976.

[Edw07] H.M. Edwards. A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, 44(3):393–422, 2007.

[Est10] N. Estibals. Compact hardware for computing the Tate pairing over 128-bit-security supersingular curves. In *Pairing-Based Cryptography–Pairing 2010*, volume 6487 of *Lecture Notes in Comput. Sci.*, pages 397–416. Springer, 2010.

[FH07] H. Fan and A. Hasan. Comments on five, six, and seven-term Karatsuba-like formulae. *IEEE Trans. Comput.*, 56(5):716–717, 2007.

[FR94] G. Frey and H.-G. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Math. Comp.*, 62(206):865–874, 1994.

[Ful69] W. Fulton. *Algebraic curves. An introduction to algebraic geometry.* Mathematics Lecture Notes Series. WA Benjamin, Inc., New York-Amsterdam. Benjamin-Cummings Publishing Co., 1969.

[Gau09] P. Gaudry. Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. *J. Symbolic Comput.*, 44(12):1690–1702, 2009.

[GGMZ13] F. Göloğlu, R. Granger, G. McGuire, and J. Zumbrägel. On the function field sieve and the impact of higher splitting probabilities: Application to discrete logarithms in $\mathbb{F}_{2^{1971}}$, 2013. Cryptology ePrint Archive Report 2013/074.

[GHS02] P. Gaudry, F. Hess, and N. P. Smart. Constructive and destructive facets of Weil descent on elliptic curves. *Journal of Cryptology*, 15(1):19–46, 2002.

[Gor93] D. M. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM Journal on Discrete Mathematics*, 6(1):124–138, 1993.

[GPR13] S. Galbraith, J. Pollard, and R. Ruprai. Computing discrete logarithms in an interval. *Math. Comp.*, 82(282):1181–1195, 2013.

[Gra08] A. Granville. Smooth numbers: computational number theory and beyond. *Algorithmic number theory: lattices, number fields, curves and cryptography, Math. Sci. Res. Inst. Publ*, 44:267–323, 2008.

[HK10] T. Helleseth and A. Kholosha. \ boldsymbol {x^{2^ l+ 1}+ x+ a} and related affine polynomials over gf (2 k). *Cryptography and Communications*, 2(1):85–109, 2010.

[HSST12] T. Hayashi, T. Shimoyama, N. Shinohara, and T. Takagi. Breaking pairing-based cryptosystems using η_t pairing over GF(3^{97}). In *Advances in Cryptology–ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Comput. Sci.*, pages 43–60, 2012.

[HSW+10] T. Hayashi, N. Shinohara, L. Wang, S. Matsuo, M. Shirase, and T. Takagi. Solving a 676-bit discrete logarithm problem in GF(3^{6n}). In *Public Key Cryptology–PKC 2010*, volume 6056 of *Lecture Notes in Comput. Sci.*, pages 351–367. Springer, 2010.

[HT93] A. Hildebrand and G. Tenenbaum. Integers without large prime factors. *J. Théor. Nombres Bordeaux*, 5(2):411–484, 1993.

[HWCD08] H. Hisil, K. Wong, G. Carter, and E. Dawson. Twisted Edwards curves revisited. In *Advances in Cryptology–ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Comput. Sci.*, pages 326–343. Springer, 2008.

[Jel12] H. Jeljeli. Accelerating iterative SpMV for discrete logarithm problem using GPUs, 2012. Preprint, available on arXiv:1209.5520.

[JL02] A. Joux and R. Lercier. The function field sieve is quite special. In *Algorithmic Number Theory–ANTS V*, volume 2369 of *Lecture Notes in Comput. Sci.*, pages 431–445. Springer, 2002.

[JL03] A. Joux and R. Lercier. Improvements to the general number field for discrete logarithms in prime fields. *Math. Comp.*, 72(242):953–967, 2003.

[JL06] A. Joux and R. Lercier. The function field sieve in the medium prime case. In *Advances in Cryptology–EUROCRYPT 2006*, volume 4005 of *Lecture Notes in Comput. Sci.*, pages 254–270. Springer, 2006.

[JL07] A. Joux and R. Lercier. Algorithmes pour résoudre le problème du logarithme discret dans les corps finis. In *Nouvelles Méthodes Mathématiques en Cryptographie*, volume Fascicule Journées Annuelles, pages 23–53. Société Mathématique de France, June 2007.

[JLSV06] A. Joux, R. Lercier, N. Smart, and F. Vercauteren. The number field sieve in the medium prime case. In *Advances in Cryptology–CRYPTO 2006*, volume 4117 of *Lecture Notes in Comput. Sci.*, pages 326–344. Springer, 2006.

[Jou00] A. Joux. A one round protocol for tripartite Diffie–Hellman. In *Algorithmic Number Theory–ANTS IV*, volume 1838 of *Lecture Notes in Comput. Sci.*, pages 385–393. Springer, 2000.

[Jou13a] A. Joux. Faster index calculus for the medium prime case application to 1175-bit and 1425-bit finite fields. In *Advances in Cryptology–EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Comput. Sci.*, pages 177–193. Springer, 2013.

[Jou13b] A. Joux. A new index calculus algorithm with complexity L(1/4 + $o(1)$) in very small characteristic, 2013. Cryptology ePrint Archive Report 2013/095.

[JV12] A. Joux and V. Vitse. Cover and decomposition index calculus on elliptic curves made practical. In *Advances in Cryptology–EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Comput. Sci.*, pages 9–26. Springer, 2012.

[K+05] T. Kleinjung et al. GGNFS, 2005. Software available at https://github.com/radii/ggnfs.

[KAF+10] T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, et al. Factorization of a 768–bit RSA modulus. In *Advances in Cryptology–CRYPTO 2010*, volume 6223 of *Lecture Notes in Comput. Sci.*, pages 333–350. Springer, 2010.

[Kob87] N. Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48(177):203–209, 1987.

[Kra22] M Kraitchik. *Théorie des nombres*, volume 1. Gauther-Villar, Paris, 1922.

[Kru10] A. Kruppa. *Speeding up Integer Multiplication and Factorization.* PhD thesis, Université Henri Poincaré - Nancy I, 2010.

[Kru11] A. Kruppa. Comparison of block-Lanczos and block-Wiedemann for solving linear systems in large factorizations, 2011. Slides available at http://event.cwi.nl/wcnt2011/slides/kruppa.pdf.

[KY06] E. Kaltofen and G. Yuhasz. On the matrix berlekamp-massey algorithm, 2006. Preprint available at http://www4.ncsu.edu/~kaltofen/bibliography/06/KaYu06.pdf.

[Len87] H. W. Jr Lenstra. Factoring integers with elliptic curves. *Annals of Mathematics*, pages 649–673, 1987.

[LL93] A. K. Lenstra and H. W. Jr Lenstra. *The development of the number field sieve*, volume 1554. Springer, 1993.

[LO91] B. A. LaMacchia and A. M. Odlyzko. Solving large sparse linear systems over finite fields. In *Advances in Cryptology–CRYPT0 '90*, volume 537 of *Lecture Notes in Comput. Sci.*, pages 109–133. Springer, 1991.

[Lor96] D. Lorenzini. *An invitation to arithmetic geometry*, volume 9 of *Graduate Studies in Mathematics*. Amer. Math. Soc., 1996.

[LP31] D. H. Lehmer and R. E. Powers. On factoring large numbers. *Bulletin of the AMS*, 37:770–776, 1931.

[LPP93] HW Lenstra, Jonathan Pila, and Carl Pomerance. A hyperelliptic smoothness test. i. *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences*, 345(1676):397–408, 1993.

[Mat99] R. Matsumoto. Using C_{ab} curves in the function field sieve. *IEICE Trans. on Fundamentals of Electronics, Communic. and Computer Sci.*, 82(3):551–552, 1999.

[Mat03] D. V. Matyukhin. On asymptotic complexity of computing discrete logarithms over GF(p). *Discrete Mathematics and Applications*, 13(1):27–50, 2003.

[MB75] M. A. Morrison and J. Brillhart. A method of factoring and the factorization of f_7. *Math. Comp.*, 29(129):183–205, 1975.

[Mil86] V. S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology–CRYPTO '85*, volume 218 of *Lecture Notes in Comput. Sci.*, pages 417–426. Springer, 1986.

[Mon92] P. L. Montgomery. *An FFT extension of the elliptic curve method of factorization*. PhD thesis, Univers. of California Los Angeles, 1992.

[Mon97] P. L. Montgomery. Square roots of products of algebraic numbers, 1997. Preliminary version, significantly different from published version.

[Mon05] P. L. Montgomery. Five, six, and seven-term Karatsuba-like formulae. *IEEE Trans. Comput.*, pages 362–369, 2005.

[Mon06] P. L. Montgomery. Searching for higher-degree polynomials for the general number field sieve, 2006. Slides available at http://www.ipam.ucla.edu/publications/scws1/scws1_6223.ppt.

[MOV93] A. J. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Inform. Theory*, 39(5):1639–1646, 1993.

[Mur99] B. A. Murphy. *Polynomial selection for the number field sieve integer factorisation algorithm*. PhD thesis, Australian National Univers., 1999.

[Neu86] J. Neukirch. *Class Field Theory*, volume 280 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1986.

[Neu99] J. Neukirch. *Algebraic Number Theory*, volume 322 of *A series of Comprehensive Studies in Mathematics*. Springer Berlin, 1999. translated by Schappacher, N.

[Ose08] I. Oseledets. Optimal Karatsuba-like formulae for certain bilinear forms in GF(2). *Linear Algebra and its Applications*, 429:2052–2066, 2008.

[PGF98] D. Panario, X. Gourdon, and P. Flajolet. An analytic approach to smooth polynomials over finite fields. In *Algorithmic Number Theory–ANTS III*, volume 1423 of *Lecture Notes in Comput. Sci.*, pages 226–236. Springer, 1998.

[PH78] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and his cryptographic significance. *IEEE Trans. Inform. Theory*, 24(1):106–110, 1978.

[Poh77] S. C. Pohlig. Algebraic and combinatoric aspects of cryptography. Technical Report 6602, Stanford Univ., 1977.

[Pol78] J. M. Pollard. Monte Carlo methods for index computation (mod p). *Math. Comp.*, 32(143):918–924, 1978.

[Pol93] J. M. Pollard. The lattice sieve. In *The development of the number field sieve*, pages 43–49. Springer, 1993.

[Pom82] C. Pomerance. Analysis and comparison of some integer factoring algorithms. *Mathematisch Centrum Computational Methods in Number Theory, Pt. 1*, pages 89–139, 1982.

[PS92] C. Pomerance and J. W. Smith. Reduction of huge, sparse matrices over finite fields via created catastrophes. *Experimental Mathematics*, 1(2):89–94, 1992.

[PZ11] T. Prest and P. Zimmermann. Non-linear polynomial selection for the number field sieve. *J. Symbolic Comput.*, 47(4):401–409, 2011.

[Rab10] F. P. Rabarison. Structure de torsion des courbes elliptiques sur les corps quadratiques. *Acta Arith.*, 144:17–52, 2010.

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21(2):120–126, 1978.

[S+11] W. A. Stein et al. *Sage Mathematics Software (Version 4.7)*. The Sage Development Team, 2011. Downloadable at http://www.sagemath.org.

[Sch93] O. Schirokauer. Discrete logarithms and local units. *Philos. Trans. Roy. Soc. London Ser. A*, 345(1676):409–423, 1993.

[Sch00] O. Schirokauer. Using number fields to compute logarithms in finite fields. *Math. Comp.*, 69(231):1267–1283, 2000.

[Sch05] O. Schirokauer. Virtual logarithms. *Journal of Algorithms*, 57(2):140–147, 2005.

[Sem02] I. Semaev. Special prime numbers and discrete logs in finite prime fields. *Math. Comp.*, 71(237):363–377, 2002.

[Ser71] J.-P. Serre. Propriétés galoisiennes des points d'ordre fini des courbes elliptiques. *Inventiones mathematicae*, 15(4):259–331, 1971.

[Ser81] J.-P. Serre. Quelques applications du théoréme de Chebotarev. *Inst. Hautes Études Sci. Publ. Math.*, 54:323–401, 1981.

[Sil09] J. H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics*. Springer, 2009.

[SLJ96] P. Stevenhagen and H.W. Lenstra Jr. Chebotarëv and his density theorem. *The Mathematical Intelligencer*, 18(2):26–37, 1996.

[Sti08] H. Stichtenoth. *Algebraic function fields and codes*, volume 254 of *Graduate Texts in Mathematics*. Springer, 2008.

[Sut12] A. Sutherland. Computing the image of Galois, 2012. Talk at the Canadian Number Theory Association XII Meeting.

[Suy85] H. Suyama. Informal preliminary report, October 1985.

[SWPD07] J R. Sendra, F. Winkler, and S. Pérez-Díaz. *Rational algebraic curves*, volume 22 of *Algorithms and Computation in Mathematics*. Springer, 2007.

[Tes01] E. Teske. On random walks for pollard's rho method. *Mathematics of computation*, 70(234):809–825, 2001.

[Tho03] E. Thomé. *Algorithmes de calcul des logarithmes discrets dans les corps finis*. PhD thesis, École Polytechnique, France, 2003.

[vzGG03] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, 2003.

[Web97] D. Weber. *On the computation of discrete logarithms in finite prime fields*. PhD thesis, Saarländische Universitäts- und Landesbibliothek, 1997.

[Z+13] P. Zimmermann et al. GMP–ECM, 2013. Downloadable at http://ecm.gforge.inria.fr/.

[ZD06] P. Zimmermann and B. Dodson. 20 years of ECM. In *Algorithmic Number Theory–ANTS VII*, volume 4076 of *Lecture Notes in Comput. Sci.*, pages 525–542. Springer, 2006.

[Zyw11] D. Zywina. On the surjectivity of mod ℓ representations associated to elliptic curves, 2011. Preprint available at http://www.math.upenn.edu/~zywina/papers/EffectiveModl.pdf.

Résumé

Dans cette thèse nous examinons en détail le problème du logarithme discret dans les corps finis. Dans la première partie, nous nous intéressons à la notion de friabilité et à l'algorithme ECM, le plus rapide test de friabilité connu. Nous présentons une amélioration de l'algorithme en analysant les propriétés galoisiennes des polynômes de division. Nous continuons la présentation par une application d'ECM dans la dernière étape du crible algébrique (NFS).

Dans la deuxième partie, nous présentons NFS et son algorithme correspondant utilisant les corps de fonctions (FFS). Parmi les améliorations examinées, nous montrons qu'on peut accélérer le calcul de logarithme discret au prix d'un pré-calcul commun pour une plage de premiers ayant le même nombre de bits. Nous nous concentrons ensuite sur la phase de sélection polynomiale de FFS et nous montrons comment comparer des polynômes quelconques à l'aide d'une unique fonction.

Nous concluons la deuxième partie avec un algorithme issu des récentes améliorations du calcul de logarithme discret. Le fait marquant est la création d'une procédure de descente qui a un nombre quasi-polynomial de nœuds, chacun exigeant un temps polynomial. Cela a conduit à un algorithme quasi-polynomial pour les corps finis de petite caractéristique.

Abstract

In this thesis we study at length the discrete logarithm problem in finite fields. In the first part, we focus on the notion of smoothness and on ECM, the fastest known smoothness test. We present an improvement to the algorithm by analyzing the Galois properties of the division polynomials. We continue by an application of ECM in the last stage of the number field sieve (NFS).

In the second part, we present NFS and its related algorithm on function fields (FFS). We show how to speed up the computation of discrete logarithms in all the prime finite fields of a given bit-size by using a pre-computation. We focus later on the polynomial selection stage of FFS and show how to compare arbitrary polynomials with a unique function.

We conclude the second part with an algorithm issued from the recent improvements for discrete logarithm. The key fact was to create a descent procedure which has a quasi-polynomial number of nodes, each requiring a polynomial time. This leads to a quasi-polynomial algorithm for finite fields of small characteristic.