Discrete logarithms in curves over finite fields

Andreas Enge

Discrete logarithms in curves over finite fields

Andreas Enge

> Logarithme, *s. m. (Arithmét.) nombre d'une progression arithmétique, lequel répond à un autre nombre dans une progression géométrique.*
> — Encyclopedia of Diderot and d'Alembert

The discrete logarithm problem in finite groups is one of the supposedly difficult problems at the foundation of asymmetric or public key cryptography. The first cryptosystems based on discrete logarithms were implemented in the multiplicative groups of finite fields, in which the discrete logarithm problem turned out to be easier than one would wish, just as the factorisation problem at the heart of RSA. The focus has then shifted towards elliptic and more complex algebraic curves over finite fields. Elliptic curves have essentially resisted all cryptanalytic efforts and to date yield the cryptosystems relying on a number theoretic complexity assumption with the shortest key lengths for a given security level, while other classes of curves have turned out to be substantially weaker. This survey presents the history and state of the art of algorithms for computing discrete logarithms in non-elliptic curves over finite fields; the case of elliptic curves is touched upon, but a thorough treatment would require an article of its own, see [**10**, Chapter V] and [**42**]. For a previous survey on hyperelliptic curves in cryptography, including the discrete logarithm problem, see [**37**].

Let us fix the notation used in the following. Given a cyclic group $(G, +)$ of order N, generated by some element P, the *discrete logarithm* of $Q \in G$ to the base P is given by the integer $x = \log Q = \log_P Q$, uniquely determined modulo N, such that $Q = xP$. The *discrete logarithm problem (DLP)* in G is to compute x given Q. A cryptosystem is said to be *based on the discrete logarithm problem in* G if computing discrete logarithms in G breaks the cryptosystem (in some specified sense). Note that it is usually unknown whether breaking the system is indeed *equivalent* to the discrete logarithm problem (but see the treatment of the computational Diffie–Hellman problem in Section 1.2).

Figure 1 illustrates the complexity of the discrete logarithm problem depending on N, as it presents itself in a number of groups suggested for cryptographic use. In the following sections, we will examine more closely algorithms in each of the

2000 *Mathematics Subject Classification.* Primary 11T71; Secondary 11Y16, 14H40.

complexity classes, going from the slower to the faster ones, that at the same time apply to a more and more restricted class of groups.

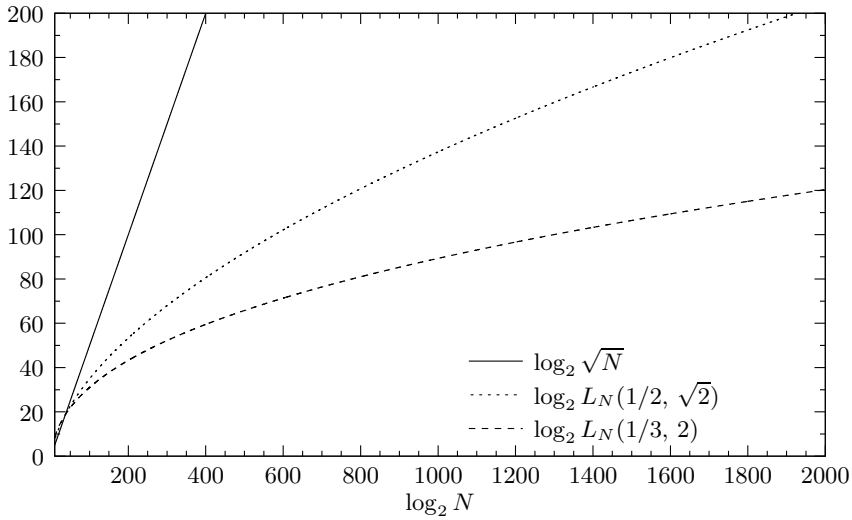


FIGURE 1. Complexity of the DLP in different groups

1. Exponential algorithms

1.1. Generic algorithms. A certain number of algorithms allows to compute discrete logarithms generically using only group operations in G, independently of the concrete representation of its elements, under the only assumption that each group element is represented canonically by a unique bit string.

Note first that the decisional version of the DLP is easy: Given Q and a candidate x for its discrete logarithm, it suffices to compute xP and to compare it to Q in order to check whether x is the correct logarithm. Hence the DLP may be solved with $O(N)$ group operations by exhaustive search.

This complexity may be reduced to $O(\sqrt{N})$. Shanks's *baby-step giant-step* algorithm [**64**] computes first the baby steps iP for $0 \leqslant i < \lceil\sqrt{N}\rceil$ and stores them in a hash table; then the giant steps $Q - j\lceil\sqrt{N}\rceil$ for $0 \leqslant j < \lceil\sqrt{N}\rceil$ are computed and looked up in the hash table containing the baby steps. As soon as a *collision* $iP = Q - j\lceil\sqrt{N}\rceil P$ occurs, the discrete logarithm $x = i + j\lceil\sqrt{N}\rceil$ is readily deduced. This deterministic algorithm performs $O(\sqrt{N})$ group operations and requires storage space for $O(\sqrt{N})$ elements.

A probabilistic approach due to Pollard [**59**] allows to dispense with the storage requirements. The basic idea is to compute random linear combinations $R_i = a_iP + b_iQ$. When a collision $R_i = R_j$ occurs, the discrete logarithm is given by $x = -\frac{a_j - a_i}{b_j - b_i} \bmod N$ if $b_j - b_i$ is invertible modulo N; otherwise, at least the partial information $x \bmod \frac{N}{\gcd(N, b_j - b_i)}$ is obtained. As such, the algorithm has an expected running time of $O(\sqrt{N})$, but still needs to store $O(\sqrt{N})$ group elements. By replacing the random choice of a_i and b_i by a pseudo-random walk such that R_{i+1} depends only on R_i, and by looking for collisions exclusively of the form

$R_i = R_{2i}$, one recovers Pollard's ϱ algorithm, that heuristically takes a time of $O(\sqrt{N})$ with constant storage. For a more advanced analysis, see [**68**].

Alternatively, one may use an approach admitting a simple parallelisation due to van Oorschot and Wiener [**57**]. First of all, *distinguished points* are defined as group elements with an easily recognisable property that occurs with a well-controlled probability, such as a certain number of zeroes in their binary representation. Several pseudo-random walks are started in parallel from different points. As soon as a distinguished point is reached, it is reported to a central machine that stores it and performs the collision search on only the stored elements.

The existence of a canonical representative for each element is crucial for the algorithms of complexity $O(\sqrt{N})$; it allows to store the elements in a hash table and to perform a search in essentially constant time. If the collision search required a test for equality with each of the stored elements, the complexity would raise again to $O(N)$.

A classical trick described in [**58**] consists in reducing the DLP to a series of discrete logarithm computations in the subgroups of order p of G for primes p dividing N. First of all, if e is the largest exponent such that $p^e|N$, one has $x \bmod p^e = \log_{(N/p^e)\cdot P}\left(\frac{N}{p^e}Q\right)$; the Chinese remainder theorem allows to compose these discrete logarithms in the Sylow subgroups of G to obtain x. So without loss of generality, we may henceforth assume that $N = p^e$ is a prime power. Similarly, $x_0 = x \bmod p$ is obtained as $\log_{p^{e-1}P}(p^{e-1}Q)$; then, $x_1 = \frac{x-x_0}{p} \bmod p$ as $\log_{p^{e-1}P}(p^{e-2}(Q - x_0P))$ and so on, so that the decomposition in base p of x is computed via a series of discrete logarithms in the subgroup of order p of G. Combined with the algorithms of square root complexity presented above, discrete logarithms may be computed with

$$O\left(\sum_{p^e||N} e\sqrt{p}\right)$$

group operations, where the sum is taken over all prime powers p^e such that $p^e|N$ and $p^{e+1} \nmid N$. So the maximal level of security reachable in a cryptosystem based on the discrete logarithm problem depends essentially on the largest prime factor of the group order. For prime N, this corresponds to the straight line in Figure 1.

1.2. Lower bounds. It would be interesting to know if a minimal difficulty of the DLP may be guaranteed. Nechaev and Shoup provide a partial answer in [**56, 65**]: If the only operations permitted are additions in the group and N is prime, then $\Omega(\sqrt{N})$ operations are needed to compute discrete logarithms with a non-negligible probability. To bypass this lower bound, an algorithm needs to take into account the particular representation of the group elements that distinguishes G from the abstract cyclic group of order N.

Let us make a digression and briefly discuss the computational Diffie–Hellman problem (CDH), that is known to be equivalent to the security of a certain number of cryptosystems. It consists in computing abP given the group elements P, aP and bP. In the same article [**65**], Shoup shows that a generic algorithm for CDH requires $\Omega(\sqrt{N})$ operations in a group of prime cardinality N. Even the decisional Diffie–Hellman problem DDH (given P, aP, bQ and a candidate Q, decide whether $Q = abP$) has the same minimal complexity in this setting.

Maurer and Wolf have shown the equivalence between CDH and DLP independently of their difficulty in [**52**]. Trivially, being able to solve DLP leads to solving CDH. For the converse direction, the authors consider the case that N is a prime such that there is an auxiliary group H, algebraic over $\mathbb{F}_N$, into which $\mathbb{F}_N$ may be embedded in a probabilistic sense (the image of an element need not exist, but in this case, a slightly perturbed element must have an image). For instance, H may be an elliptic curve defined over $\mathbb{F}_N$, and the image of $x \in \mathbb{F}_N$ is given by a point on H with abscissa x, if it exists; otherwise, one may continue with $x + e$ for a small integer e. If in this situation the DLP in H can be solved by an algebraic algorithm carrying out n group operations, the DLP in G can be solved by an algorithm making essentially n calls to an oracle for CDH in G. If the order of H is sufficiently smooth (that is, it does not have prime factors exceeding a polynomial in $\log N$), the algorithms of Section 1.1 applied to H lead to a polynomial time equivalence between CDH and DLP in G. Notice that only the order of G plays a role in this argument, but not the concrete representation of its elements. Using the fact that all integers in the Hasse interval around N appear as cardinalities of elliptic curves over $\mathbb{F}_N$ and assuming heuristically that they have the same factorisation pattern as random integers of the same size, Maurer and Wolf show the existence of an auxiliary group H such that the reduction becomes polynomial. Finding the group via complex multiplication, however, may take exponential time. If a subexponential reduction in $L(1/2)$ is considered sufficient instead of a polynomial reduction, it should be possible to find the group in the same subexponential time.

The generic, exponential algorithms are for the time being the only ones that may be applied to arbitrary elliptic curves. Some particular elliptic curves admit an embedding into another group in which discrete logarithms are considerably easier to compute, but these curves have a very low density: They are supersingular and other curves with a low embedding degree into the multiplicative group of a finite field [**53, 32**]; subgroups of order p defined over a finite field $\mathbb{F}_q$ of characteristic p, that may be embedded into the additive group $(\mathbb{F}_q, +)$ [**61, 62, 66**]; and elliptic curves that may be embedded by Weil descent into the Jacobian of a hyperelliptic curve of low genus following an idea suggested by Frey in [**31**], see [**34, 18**] and [**42**] and the references therein. Weil descent provides another motivation for examining more closely curves of genus larger than 1.

2. Subexponential algorithms of complexity $L(1/2)$

2.1. The subexponential function. Under the designation *subexponential function*, one might subsume all functions that grow more slowly than exponentially, but faster than polynomially. In the context of discrete logarithm or factorisation algorithms, the following more restrictive definition appears naturally:

Definition 1. The *subexponential function* with parameters $\alpha \in (0, 1)$ and $c > 0$ with respect to the argument N is given by

$$L_N(\alpha, c) = e^{c(\log N)^\alpha (\log\log N)^{1-\alpha}}.$$

To simplify the notation, we let

$$L_N(\alpha) = \{L_N(\alpha, c) : c > 0\}$$

and omit the subscript N when it is understood from the context.

In the following, we will focus on the parameter α, that has the biggest influence on the growth of the function. The parameter c is often called the *constant* of the subexponential function, although it appears in fact in the exponent, so that its influence is far from negligible.

The traditional notation L_N may lead to confusion, as in terms of complexity, one has to assume that a problem with N possible inputs is specified by $\log N$ bits, and subexponentiality has to be understood with respect to $\log N$ rather than N:

- The extreme case $\alpha = 0$, excluded by the definition, leads to the polynomial $\log^c N$;
- the other extreme case $\alpha = 1$ leads to the exponential N^c;
- as intermediate values, essentially $\alpha = 1/2$ and $\alpha = 1/3$ occur in the contexts of discrete logarithms and of factorisation. Two typical functions are traced in Figure 1.

The following computation rules are easily checked:

$$\begin{aligned} &L_N(\alpha, c_1) \cdot L_N(\alpha, c_2) = L_N(\alpha, c_1 + c_2) \\ &\log^k N \in L(\alpha, o(1)) \text{ for any } k, \text{ and more generally,} \\ &L_N(\beta, d) \in L_N(\alpha, o(1)) \text{ for } \beta < \alpha. \end{aligned} \tag{1}$$

In particular, if a polynomial time operation is repeated $L_N(\alpha, c)$ times, the resulting complexity is in $L_N(\alpha, c + o(1))$; this is why Definition 1 is often modified to allow a $o(1)$ term in the constant.

2.2. An algorithm for finite fields. Subexponential algorithms for discrete logarithms usually proceed in two stages: In the first stage, called *sieving* or *relation collection*, an integral matrix is filled with *relations*; the *linear algebra* stage solves the resulting system modulo the group order and yields the discrete logarithms of certain elements; a third, comparatively inexpensive stage may be needed to compute *individual logarithms*. It has become common to call this kind of algorithm "index calculus", a rather unfortunate terminology, since "index" is traditionally used as a synonym for "logarithm". Already the encyclopedia by Diderot and d'Alembert, published between 1751 and 1772, gives the following definition: "*Index*, en terme d'Arithmétique, est la même chose que la caractéristique ou l'exposant d'un logarithme. Voyez Logarithme." [**17**].

The basic idea of creating relations and of combining them linearly for computing discrete logarithms (and for factoring) has been published by Kraïtchik in the twenties [**46**, Chapter 5, §§14-16]. In 1979, the algorithm has been rediscovered by Adleman and presented with the analysis of its subexponential complexity for the case of finite prime fields. It is easily generalised to $\mathbb{F}_{2^m}$ (for the reasons explained in Section 2.5). In the following, we describe a slightly modified version.

To recall the problem, let P be a primitive element of $\mathbb{F}_{2^m}$ and $Q \in \mathbb{F}_{2^m}^\times$; we wish to compute x such that $Q = P^x$. The finite field is conveniently represented as $\mathbb{F}_2[X]/(f)$ with f an irreducible polynomial of degree m, such that an element of $\mathbb{F}_{2^m}$ may be considered as a binary polynomial of degree less than m. This representation of the field elements by polynomials introduces notions that in principle have no meaning in a field: It is now possible to speak of *irreducible elements*, the degree of the polynomials leads to a notion of *size* of the elements, and there is a *unique factorisation* of elements into irreducibles. In fact, the factorisation is no more

unique as soon as the restriction on the degrees is lifted, as several elements of $\mathbb{F}_2[X]$ may represent the same element of the finite field.

ALGORITHM 2.
INPUT: P a primitive element of $\mathbb{F}_{2^m} = \mathbb{F}_2[X]/(f)$, $Q \in \mathbb{F}_{2^m}^\times$
OUTPUT: x such that $Q = P^x$

(0) Let $N = 2^m - 1$. Fix a *smoothness bound* $B \in \mathbb{N}$ and compute the *factor base* $\mathcal{F} = \{p_0, \ldots, p_n\}$ containing the irreducible polynomials over $\mathbb{F}_2$ of degree at most B and $P = p_0$. Prepare an empty matrix A with n columns and r rows and an empty vector b with r rows for some r slightly larger than n.
(1) **repeat** for $i = 1, \ldots, r$
  repeat
    draw random exponents $e_{ij} \in \{0, \ldots, N-1\}$ for $j = 0, \ldots, n$
    compute $\prod_{j=0}^n p_j^{e_{ij}} \bmod f$
    if the result factors over $\mathcal{F}$ as $\prod_{j=0}^n p_j^{f_{ij}}$
      there is a *relation* $\prod_{j=1}^n p_j^{a_{ij}} = P^{-a_{i0}}$ in $\mathbb{F}_{2^m}$ with $a_{ij} = e_{ij} - f_{ij}$
      add $(a_{ij})_{j=1}^n$ to the matrix A and $-a_{i0}$ to the vector b
  until success in creating a new relation
(2) Solve the linear system $Ay = b$ modulo N, so that $y_j = \log_P p_j$.
(3) Create an additional relation $Q \prod p_j^{e_j} = \prod p_j^{f_j}$ as above; return $x = \sum (f_j - e_j) y_j$.

This version of the algorithm separates the linear algebra of stage 2 and the computation of individual logarithms of stage 3, that may be repeated as many times as desired. Alternatively, it would be enough to add the target Q to the factor base and to stop after stage 2.

The complexity of the algorithm depends essentially on the probability that a polynomial of degree at most $m-1$ decomposes completely over the factor base, otherwise said, that it is *B-smooth*. If the factor base size n is polynomial, this probability decreases exponentially; for it to decrease polynomially, one would need n to be exponential. The optimal value is somewhere in between; precisely, for B chosen such that $n \in L(1/2)$, the probability of obtaining a relation is in $1/L(1/2)$ (see [**9**] and Section 2.5). Hence, the expected number of iterations for obtaining one relation is in $L(1/2)$, and the process has to be repeated $r \in O(n) \subseteq L(1/2)$ times to fill the matrix. As all the basic operations of the algorithm are polynomial or in $L(1/2)$ (for instance, the linear algebra stage is polynomial in n), the computation rules (1) show that the total complexity of the algorithm is in $L(1/2)$. Smoothness is discussed in more generality and a more detailed complexity analysis is developed in Section 2.5.

Notice that the subexponential complexity of Algorithm 2 does not contradict the exponential lower bounds of Section 1.2: We clearly make use of the particular representation of the elements of $\mathbb{F}_{2^m}^\times \simeq \mathbb{Z}/N\mathbb{Z}$ by polynomials, and the algorithm is far from generic.

2.3. Arithmetic of Jacobians. In the light of Section 1.2, it is clear that we need to take a closer look at the representation of elements and at the group law

associated to an algebraic curve. To arrive at the analogue of Algorithm 2, our aim is to show that these elements behave essentially like polynomials.

Curves over algebraically closed fields. For the time being, let us consider a curve $\mathcal{C}$ defined by a non-singular irreducible polynomial $C(X, Y)$ over an *algebraically closed* field K. The *points* on $\mathcal{C}$ are the $(x, y) \in K^2$ such that $C(x, y) = 0$. Non-singularity means that for no point (x, y) on the curve, the partial derivatives $\frac{\partial C}{\partial X}(x, y)$ and $\frac{\partial C}{\partial Y}(x, y)$ vanish simultaneously. There is an integer g, called the *genus* of the curve, that is closely related to the degree of C if the equation is "reasonable", and that measures, roughly speaking, how "complicated" the curve is. For instance, a *hyperelliptic* curve in characteristic different from 2 is given by a non-singular polynomial $C = Y^2 - X^{2g+1} - f(X)$ with f of degree at most $2g$; the case $g = 1$ is that of an *elliptic* curve. A more general case is that of *superelliptic* curves, defined by a non-singular polynomial $Y^a - X^b - f(X)$ with f of degree less than b and $\gcd(a, b) = 1$ when the characteristic of K is coprime with a. By admitting certain mixed terms, one obtains the most general curves that have been suggested for use in cryptography, namely $\mathcal{C}_{a,b}$ curves, given by a non-singular irreducible polynomial of the form

$$Y^a - X^b - \sum_{(i,j):ai+bj<ab} c_{ij} X^i Y^j$$

with $\gcd(a, b) = 1$ when the characteristic of K divides neither a, nor b. The genus of these curves is given by $g = \frac{(a-1)(b-1)}{2}$.

To a curve, one can associate its *coordinate ring* $K[C] = K[X, Y]/(C)$, the ring of polynomial functions from the curve to the field K. In the case of a $\mathcal{C}_{a,b}$ curve, $K[\mathcal{C}]$ can be seen as the set of polynomials of arbitrary degree in X and of degree at most $a-1$ in Y, since each occurrence of Y^a may be replaced by $X^b + \sum c_{ij} X^i Y^j$. The field of fractions of $K[\mathcal{C}]$ is denoted by $K(\mathcal{C})$ and is called the *function field* of $\mathcal{C}$; it consists of the rational functions from the curve to K.

Except for elliptic curves, the associated group does not consist of only the points on the curve. Instead, one has to consider the *Jacobian* $\mathrm{J}(\mathcal{C})$ of the curve, an abelian variety. In practice, it is preferable to work with the isomorphic group (denoted by $\mathrm{Pic}^0(\mathcal{C})$ or again by $\mathrm{J}(\mathcal{C})$) of divisor classes of degree 0. Define the group of *divisors* of $\mathcal{C}$ by

$$\mathrm{Div}(\mathcal{C}) = \left\{ \sum_{P \in \mathcal{C}} m_P P : m_P \in \mathbb{Z}, \text{ almost all zero} \right\},$$

the set of finite formal sums of points with potentially negative coefficients. This definition is in fact slightly wrong; instead of only considering points on the affine curve, one needs to also take into account "points at infinity" on the projective closure of $\mathcal{C}$. Moreover, the projective closure will usually be singular at infinity; instead of a singular point, one needs to consider several points corresponding to its resolution on a non-singular model. Equivalently, one may define divisors as formal sums of places of the function field $K(\mathcal{C})$ instead of points. Function fields of hyperelliptic or, more generally, $C_{a,b}$ curves are particular in that they have only one place at infinity; so it suffices to augment the set of points by one additional special point called ∞.

The *degree* of a divisor is given by

$$\deg\left(\sum m_P P\right) = \sum m_P.$$

The degree 1 divisors containing only one point with multiplicity 1 are called *prime*; indeed, they constitute indivisible atoms, and any divisor may be written uniquely as a sum of prime divisors.

Associate to a rational function $f \in K(\mathcal{C})$ its *principal divisor*, containing its zeroes with positive and its poles with negative multiplicities. At ∞ on a $C_{a,b}$ curve, the function X has a pole of order b, the function Y a pole of order a; the order at ∞ of any other function may be deduced by the triangular inequality. It turns out that the degree of a principal divisor is 0, otherwise said, a rational function has as many zeroes as poles, counting multiplicities. Let $\mathrm{Div}^0(\mathcal{C})$ denote the group of degree 0 divisors and $\mathrm{Prin}(\mathcal{C})$ its subgroup of principal divisors; then the Jacobian is given by

$$\mathrm{J}(\mathcal{C}) = \mathrm{Div}^0(\mathcal{C}) / \mathrm{Prin}(\mathcal{C}).$$

Given ∞ (or any other point on the curve, for that matter), there is a natural isomorphism

$$\mathrm{Div}^0(\mathcal{C}) \to \mathrm{Div}'(\mathcal{C}), \quad \sum_P m_P P \mapsto \sum_{P \neq \infty} m_P P,$$

where $\mathrm{Div}'(\mathcal{C})$ denotes the subgroup of $\mathrm{Div}(\mathcal{C})$ of divisors not containing ∞ in their support. The inverse isomorphism is obtained by adding the right multiple of ∞ to obtain a degree 0 divisor. By the Riemann–Roch theorem, each class of $\mathrm{J}(\mathcal{C})$ can then be represented by a unique *effective* or *positive* divisor (that is, without negative coefficients) in $\mathrm{Div}'(\mathcal{C})$ of minimal degree, which is called its *reduced (along ∞) representative*. Its degree is moreover bounded by the genus g.

The coordinate ring $K[\mathcal{C}]$ is in fact the set of functions without poles at infinity (or otherwise said, the integral closure of $K[X]$ in $K(\mathcal{C})$). But since for $\mathcal{C}_{a,b}$ curves, ∞ is the only point at infinity, this implies that the affine points on the curve are in bijection with the prime ideals of $K[\mathcal{C}]$, that $\mathrm{Div}'(\mathcal{C})$ is isomorphic to the group of fractional ideals of $K[\mathcal{C}]$ and that $\mathrm{J}(\mathcal{C})$ is isomorphic to the ideal class group of $K[\mathcal{C}]$. This observation allows to switch to the standard representation of ideals in extensions of Dedekind domains: Any divisor $D \in \mathrm{Div}'(\mathcal{C})$ may be represented by an ideal of $K[\mathcal{C}]$ in the form

$$D = (d)(u, w) \tag{2}$$

with d, $u \in K[X]$ monic and $w \in K[\mathcal{C}]$ (cf. [**22**, § 163, p. 461] or [**51**, Th. 17] for a proof in the number field case). The polynomial w may be taken to be monic and, for a $\mathcal{C}_{a,b}$ curve, of degree less than a in Y. Since (d) is principal, any element of the Jacobian is represented as (u, w). Even without recourse to the theory of Dedekind rings, the existence of such a representative may be shown by choosing $u \in K[X]$ having as zeroes (with the right multiplicities) the X-coordinates of the points in the divisor D, and by letting the bivariate polynomial w interpolate (again with the correct multiplicities, which requires some care) the Y-coordinates. Notice that a prime divisor $P = (x, y)$ is characterised by a representative, namely $(X - x, Y - y)$, in which the first polynomial is irreducible.

Relying on the representation (2) of divisors, the algorithm realising the group law in a Jacobian works with polynomials and proceeds in two steps: The *composition* step corresponds to the addition of the divisors respectively the multiplication of the ideals, while taking out principal ideals (d) of $K[X]$ that may appear; essentially, this is Lagrangian interpolation. The *reduction* step computes for the resulting divisor, that is generically of degree $2g$, its unique reduced representative

of degree at most g; this part of the algorithm depends heavily on the curve. Efficient algorithms for arbitrary curves have been developed by Heß and Khuri-Makdisi [**40, 44**]. For hyperelliptic curves, see, for instance, [**12, 23, 48, 38**]; for superelliptic curves, see [**33, 8**], for $\mathcal{C}_{a,b}$ and in particular $\mathcal{C}_{3,4}$ curves, see [**4, 5, 30, 7, 1**].

Curves over finite fields. In order to obtain finite groups, it is clearly necessary that the curves under consideration be defined over a finite field $K = \mathbb{F}_q$. Contrarily to what one might expect, it is not sufficient to emulate the construction made for algebraically closed fields: When adding two elements containing only points defined over $\mathbb{F}_q$, the reduction may result in a divisor containing points defined over an extension field. To save the situation, one needs to resort to Galois invariance and consider the Frobenius automorphism of $\mathbb{F}_q$, $x \mapsto x^q$, that yields naturally the endomorphism $\varphi : (x, y) \mapsto (x^q, y^q)$ on the points of the curve defined over $\overline{\mathbb{F}}_q$. It is trivially extended to divisors and rational functions. The groups Div, Div^0 and Prin may thus be defined as above as sets of divisors defined over $\overline{\mathbb{F}}_q$, but with the additional restriction that they be invariant under φ. To obtain Div' in an analogous manner, one furthermore needs ∞ to be defined over $\mathbb{F}_q$, which is the case for all curves under consideration. So once again, we end up with the ideal class group of $K[\mathcal{C}]$. The elements of the Jacobian are represented as above by ideals (u, w), now with u and w having coefficients in $\mathbb{F}_q$. The algorithms of composition and reduction remain unchanged; their algebraic nature implies that they have no "conscience" of the field over which they work.

By Weil's theorem [**70**], the order of the Jacobian of a curve $\mathcal{C}$ of genus g defined over $\mathbb{F}_q$ satisfies

$$(\sqrt{q} - 1)^{2g} \leqslant |\,\mathrm{J}(\mathcal{C})| \leqslant (\sqrt{q} + 1)^{2g}. \tag{3}$$

Composition and decomposition. The previous discussion shows that the arithmetic of the Jacobian groups of the curves under consideration boils down to that of bivariate polynomials. But as far as discrete logarithms are concerned, the group elements even behave essentially like univariate polynomials.

As a consequence of Weil's theorem, the majority of elements of the Jacobian is represented by (u, w) with $\deg u = g$, $w = Y - v(X)$ and $\deg v = g - 1$. When two distinct elements $D_1 = (u_1, Y - v_1)$ and $D_2 = (u_2, Y - v_2)$ are to be added, with overwhelming probability one has $\gcd(u_1, u_2) = 1$, or otherwise said, the points in D_1 have distinct X-coordinates from those in D_2. Then the result of the composition is $D_1 + D_2 = (u, Y - v)$ with $u = u_1 u_2$ and v the Lagrangian interpolation polynomial such that $v_i = v \bmod u_i$, which is in fact independent of the curve. The composition step for doubling a divisor $(u_1, Y - v_1)$ usually results in $(u, Y - v)$ with $u = u_1^2$ and v a Hensel lift (that depends on the curve). As long as $\deg u$ does not exceed g, in general no reduction occurs. Otherwise, the reduction step is also specific to the curve. So adding divisors corresponds to multiplying the u-polynomials in $\mathbb{F}_q[X]$ and updating the v-polynomials accordingly, followed by a reduction step. In this sense, the addition in the Jacobian behaves like multiplication in $\mathbb{F}_q^{g+1}$, which also proceeds by multiplying polynomials of degree at most g, followed by a reduction.

Over a finite field $\mathbb{F}_q$, a *prime divisor* is a divisor that cannot be written as a sum of two non-trivial divisors defined over the same field. Concretely, a prime divisor of degree k is given by the orbit D under the Galois endomorphism of a point $P = (x, y)$ with coordinates x and y in $\mathbb{F}_{q^k}$, but not both in the same subfield. A

typical representative occurs when x itself is not defined in a subfield of $\mathbb{F}_{q^k}$; then $D = (u, Y - v)$ with u the minimal polynomial of degree k of x. (The other prime divisors have the form (u, w) with $\deg u$ a proper divisor of k and $\deg_Y w = k/\deg u$ and occur in negligible numbers.) In any case, prime divisors are again characterised by having an irreducible first polynomial.

As the decomposition of an element of $\mathbb{F}_{q^{g+1}}$ in Algorithm 2, the prime decomposition of a divisor of the form $D = (u, Y - v)$ also boils down to factoring a polynomial in $\mathbb{F}_q[X]$; if $u = \prod u_i^{e_i}$, then $D = \sum e_i D_i$ with $D_i = (u_i, Y - v \bmod u_i)$.

2.4. Algorithms for hyperelliptic curves. The first subexponential algorithm for computing discrete logarithms in hyperelliptic curves of large genus defined over a finite field $K = \mathbb{F}_q$ is due to Adleman, DeMarrais and Huang [**3**]. It differs from Algorithm 2 essentially by the way in which the relations are created. Let the factor base $\mathcal{F}$ be given by all prime divisors of degree bounded by some smoothness bound B. Since principal divisors are zero in the Jacobian, it is sufficient to draw random polynomials of the form $Y - v(X)$ and to compute their divisors (higher degrees in Y do not occur in hyperelliptic curves, since the polynomials may be reduced modulo the curve equation of degree 2 in Y). A smooth divisor, that is a divisor decomposing over $\mathcal{F}$, directly yields a relation. This is the case if the norm of $Y - v$ with respect to the function field extension $K(\mathcal{C})/K(X)$, a polynomial in X, is B-smooth. Assuming heuristically that norms behave like random polynomials of the same degree, the authors prove a complexity of $L_{q^g}(1/2)$ whenever $(2g + 1)^{0.98} \geqslant \log q$. The result is heuristic for a second reason. Implicitly, Algorithm 2 describes an isomorphism between the group and $\mathbb{Z}^n$ modulo the lattice formed by the rows of the relation matrix. It is unclear whether the bounds one needs to impose on the degree of v for a subexponential running time allow to obtain a sufficiently dense lattice to yield the isomorphism.

The first algorithm for discrete logarithms in hyperelliptic curves with a proven subexponential running time is given in [**25**]. Essentially, it is Algorithm 2, that applies directly to curves via the discussion at the end of Section 2.3. It relies on the fact, proved in [**28**], that the proportion of smooth divisors is the same as the proportion of smooth univariate polynomials. (A similar result for the discrete logarithm problem in the infrastructure of a real quadratic function field can already be found in [**54**]; it also relies on the smoothness theorem of [**28**].) The constant of the subexponential complexity depends on the growth of the genus g with respect to the finite field size q; precisely, a running time of

$$L_{q^g}\left(1/2, \frac{5}{\sqrt{6}}\left(\sqrt{1 + \frac{3}{2\vartheta}} + \sqrt{\frac{3}{2\vartheta}}\right) + o(1)\right)$$

is proved in [**25**] under the assumption that $g \geqslant \vartheta \log q$.

2.5. A general framework. The similarities between finite fields, Jacobians of curves and other groups in which subexponential algorithms in $L(1/2)$ exist to solve the discrete logarithm problem, have motivated us to develop a framework that allows an abstract presentation and unified analysis independently of the group [**26**]. It is explained in the following to give a more detailed complexity analysis for Jacobians of curves, as it is not more involved than a treatment of only the curve case.

Let $\mathcal{P}$ be a set of elements called *prime*, and let $\mathbb{M}$ be the free monoid over $\mathcal{P}$. Suppose there is an equivalence relation $\sim$ in $\mathbb{M}$, compatible with addition, such that $G = \mathbb{M}/\sim$ is a group. Suppose furthermore that there is a *size function* $\deg : \mathcal{P} \to \mathbb{R}^{\geqslant 1}$ (extended homomorphically to $\mathbb{M}$), which allows to define the factor base $\mathcal{F}$ as the set of prime elements of size bounded by a smoothness bound B. If the elements of G have canonical representatives in $\mathbb{M}$ (usually, the smallest ones), the unique decomposability of elements of $\mathbb{M}$ into primes is inherited by G. If some technical conditions on G concerning, for instance, the computability of the group law, the bit size of elements or the generation of G by $\mathcal{F}$, are satisfied, then Algorithm 2 may be applied without modification to G.

These notions have been introduced by Knopfmacher in [**45**]; he calls $\mathbb{M}$ an *arithmetical semigroup* and G an *arithmetical formation*. Concrete examples are provided by prime fields $\mathbb{F}_p$, for which $\mathbb{M} = \mathbb{Z}$, deg is the logarithm and $\sim$ is equivalence modulo p; and finite fields $\mathbb{F}_{p^m} = \mathbb{F}_p[X]/(f)$, for which $\mathbb{M} = \mathbb{F}_p[X]$, deg is indeed the degree and $\sim$ is equivalence modulo f. But also class groups of number fields, for which $\mathbb{M}$ is the set of integral ideals, deg is the logarithm of the norm and $\sim$ is equivalence modulo prime ideals. And finally Jacobians of curves $\mathcal{C}$ over a finite field $\mathbb{F}_q$ with a unique point at infinity, for which $\mathbb{M} = \mathrm{Div}'(\mathcal{C})$, deg is the degree of a divisor and $\sim$ is equivalence modulo principal divisors.

It remains to prove the running time of the algorithm. For it to be in $L(1/2)$, we need that a factor base of size $L(1/2)$ implies a smoothness probability of $1/L(1/2)$. Corresponding results can be found, for instance, in [**60**] for $\mathbb{F}_p$, in [**9**] for $\mathbb{F}_{2^m}$, in [**63**] for class groups of imaginary-quadratic number fields (under the generalised Riemann hypothesis) and in [**28**] for hyperelliptic curves of large genus. Having a closer look at these smoothness theorems, one realises that they are essentially all the same: For a factor base of size $L_N(1/2, c)$, an element of size $\log N$ is smooth with a probability of $1/L_N(1/2, 1/(2c) + o(1))$. This result may be proved in $\mathbb{M}$ under an assumption analogous to the prime number theorem: The number of primes of size bounded by k must be of the order of $\frac{q^k}{k}$ for some q, see [**50, 49**]. Then the number of elements of size at most x that are smooth with respect to a bound y is asymptotically (with some constraints on the respective growth of x and y) given by the value of the Dickmann–de Bruijn function ϱ in $u = \frac{x}{y}$; and de Bruijn has shown in [**11**, (1.8)] that $1/\varrho(u) \in e^{(1+o(1))u \log u}$, which provides the link with the subexponential function.

Due to the equivalence relation, the smoothness result for $\mathbb{M}$ cannot be directly transferred to G. In a curve, for instance, there are non-reduced divisors of degree g, that as such do not occur as representatives of Jacobian elements. Nevertheless, the results of [**63, 28**] provide examples of arithmetical formations in which the same smoothness behaviour may be observed; it is thus reasonable to accept it heuristically also in other contexts.

Given the smoothness result, the complexity of Algorithm 2 may be easily verified. Let $n = L_N(1/2, d)$ denote the size of the factor base, with N the group order and d a parameter to be determined later; in the curve case, N must be replaced by q^g, which makes sense in the light of Weil's theorem (3). If a group element may be decomposed over the factor base in time $L_N(1/2, o(1))$ (which is the case for all groups under consideration), the time to create $r = O(n)$ relations in the first stage of the algorithm is in

$$L_N(1/2, o(1)) \cdot L_N(1/2, 1/(2d) + o(1)) \cdot L_N(1/2, d) = L_N(1/2, d + 1/(2d) + o(1))$$

by (1). The linear algebra stage treats a sparse matrix of order $L_N(1/2, d)$ with $L_N(1/2, d + o(1))$ entries, as each relation has on the order of $\log N$ coefficients. The Lanczos and Wiedemann algorithms of [**47, 71**] run in time $L_N(1/2, 2d+o(1))$ on these matrices. Hence, the total running time of the algorithm becomes

$$L_N(1/2, \max(d + 1/(2d), d) + o(1)).$$

This quantity is minimised by $d = \sqrt{2}/2$, resulting in a complexity of

$$L_N(1/2, \sqrt{2} + o(1)).$$

For hyperelliptic curves, this running time holds when the field size q remains fixed and the genus g tends to infinity. When q grows as well, the discrete nature of the degree function starts to play a role, but a moderate growth of q may be tolerated (see also Section 2.6); following the analysis of [**25**], a running time of

$$L_N\left(1/2, \sqrt{2} + \frac{2}{\vartheta} + o(1)\right)$$

is obtained in [**26**] for the case that $g \geqslant \vartheta \log q$.

An assumption that is implicit in Algorithm 2 need not be satisfied for curves: The group must by cyclic and of known order N. If this is not the case, one may replace the solution of a linear system by the computation of the Hermite and Smith normal forms of the matrix, which yields a complexity of $L_N(1/2)$ with a worse constant as, for instance, at the end of Section 2.4; see [**24**].

An algorithm of proved subexponential complexity in $L_{q^g}(1/2 + \varepsilon)$ is given by Couveignes in [**16**] for a large class of curves, not limited to hyperelliptic ones, under the mild assumption that the curve contains an $\mathbb{F}_q$-rational point and that the order of its Jacobian is bounded by $q^{g+O(\sqrt{g})}$. The approach is quite different from the one presented here and relies on a double randomisation, of the combination of factor base elements as well as of the choice of a function in a certain Riemann–Roch space. An algorithm without restriction on the input curve is given by Heß in [**41**], who thus proves a complexity of $L_{q^g}(1/2)$ for all curves of large genus.

2.6. The low genus case. At first sight, these algorithms do not seem to work in low genus. This is nicely illustrated by the case of elliptic curves, which are of genus 1, so that each reduced divisor contains exactly one point: Either, the smoothness bound is set to $B = 1$, in which case any divisor is smooth, and the matrix contains as many columns as there are elements in the group, that is around q. So the algorithm must be slower than the generic ones of Section 1.1, of complexity $O(\sqrt{q})$. Or the smoothness bound is set to $B = 0$, in which case no divisor is smooth. The problem stems again from the discreteness of B, that is smoothed out when the genus becomes larger.

In fact, interesting results are already obtained for rather small genus, as first observed by Gaudry in [**36**]. Assume g to be fixed, while q tends to infinity. Choose a smoothness bound of $B = 1$, so that the factor base is composed of divisors containing only one point. By the Weil bound, its size n satisfies $|n - (q + 1)| \leqslant 2g\sqrt{q} = O(\sqrt{q})$. The smooth reduced divisors are essentially the multisets containing g points (cf. Section 2.3); asymptotically for $q \to \infty$, multiplicities do not play a role, so that the number of smooth reduced divisors is well approximated by $\binom{n}{g} = \frac{q^g}{g!} + O(q^{g-1/2})$. As by (3) the Jacobian group has a size

of $q^g + O(q^{g-1/2})$, the smoothness probability for a random element is, asymptotically for $q \to \infty$, given by $\frac{1}{g!}$. So filling the matrix requires $g!n = O(q)$ trials, and the linear algebra step on a sparse matrix with $O(n)$ rows and columns and g entries per row takes $O(n^2 g) = O(q^2)$ arithmetic operations. While this complexity is exponential in the group size q^g, the generic algorithms of complexity $q^{g/2}$ are beaten as soon as $g \geqslant 5$. Notice that the two stages of the algorithm are quite unbalanced; by reducing the factor base size, an idea attributed to Harley in [**36**], one may slow down the relation collection process while speeding up the linear algebra. Letting $n = O(q^r)$ with $r = 1 - 1/(g+1)$, the total complexity becomes $O(q^{2-2/(g+1)})$ arithmetic operations, which is better than the generic complexity already for $g = 4$.

A further improvement is given by Thériault in [**69**]. Again, the factor base comprises only a fraction of the prime divisors of degree 1, chosen arbitrarily, say $n = q^r$ with some parameter r to be optimised. The other prime divisors of degree 1, however, are not discarded any more, but form the set of *large primes* (that in this context, of course, are not larger than the others, but the terminology as well as the basic idea is inspired by the large prime variation in the factorisation context). A relation is retained if it either consists of prime divisors in the factor base (the case of a *full relation*) or if it contains exactly one large prime besides elements of the factor base (the case of a *partial relation*). Before entering the linear algebra step, k partial relations with the same large prime are combined to form $k-1$ full relations; large primes that occur only once are eliminated. The net effect is similar to the choice of a smaller factor base: Relation collection is slowed down (but not as much), while the linear algebra is accelerated. Thériault shows that the optimal value for r is $1 - 2/(2g+1)$ for a final running time of $O(q^{2-4/(2g+1)})$ arithmetic operations; this is slightly better than the generic algorithms already for $g = 3$.

Finally, Gaudry, Thomé, Thériault and Diem in [**35**] and Nagao in [**55**] have suggested the use of two "large" primes, which complicates the process of recombining partial relations, but allows to reduce the factor base size even further. The optimal value $r = 1 - 1/g$ yields a running time of $O(q^{2-2/g})$ arithmetic operations.

The above algorithms are formulated for Jacobians of hyperelliptic curves, but carry over to arbitrary curves. One conclusion to draw might be that curves of genus 3 and above should be banned from cryptography, as they are less secure than lower genus curves for the same group order. As a more nuanced reaction, one may also decide to increase the group size slightly, especially for a genus close to the cross-over point. In genus 3, for instance, one would need to increase the bit length of the group order by 12.5 % for an equivalent level of security compared to elliptic or genus 2 curves. This need not be penalising since machine word sizes introduce an effect of discretisation into the implementation.

3. Subexponential algorithms of complexity $L(1/3)$

Following the progress for factorisation algorithms, a complexity of $L(1/3)$ has also been established for discrete logarithm computations in finite fields. First of all, Coppersmith's algorithm [**15**] treats $\mathbb{F}_{2^m}$; it may be seen as a special case of Adleman's *function field sieve* [**2**], that applies to fields $\mathbb{F}_{p^m}$ with p small. The case of $\mathbb{F}_p$ respectively $\mathbb{F}_{p^m}$ for m small is handled by Gordon's *number field sieve* [**39**]. Recently, it has been shown in [**43**] that the applicability domains of the two

algorithms intersect, so that a complexity of $L(1/3)$ is obtained for arbitrary finite fields.

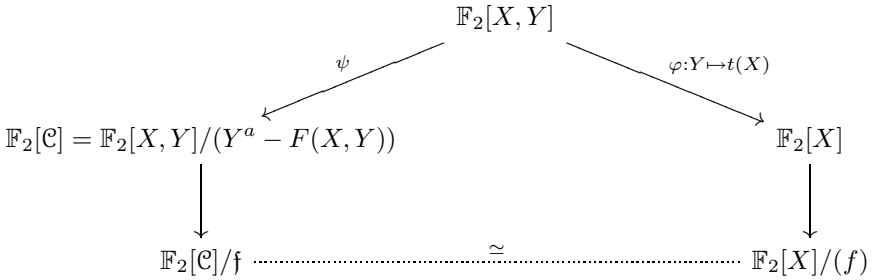
3.1. The function field sieve. The function field sieve is of particular interest in our case since it is related to the algorithm for curves of Section 3.2. Its starting point is the observation that the smoothness results of Section 2.5 may be generalised by varying the size of the elements to be tested for smoothness and the smoothness bound. The following theorem is proved in [**27**] for algebraic curves whose genus grow sufficiently fast compared to a power of $\log q$, but it holds in more generality.

THEOREM 3. *Given an arithmetical formation of order N as in Section 2.5 in which smoothness is governed by the Dickmann–de Bruijn function, let $0 < \beta < \alpha \leqslant 1$ and c, $d > 0$. The probability that an element of size at most $\log L_N(\alpha, c)$ is smooth with respect to the factor base containing the $L_N(\beta, d)$ smallest primes is given by*

$$1/L_N\left(\alpha - \beta, (\alpha - \beta)\frac{c}{d} + o(1)\right).$$

The special case $\alpha = c = 1$ and $\beta = 1/2$ has been used in the previous section to prove complexities in $L(1/2)$. To reach a complexity of $L(1/3)$, this theorem opens only one direction: Since the factor base has to be written down, one may not exceed $\beta = 1/3$, whence the size of the elements to be decomposed has to be lowered to $\log L(2/3)$.

The function field sieve succeeds in this goal by representing the finite field $\mathbb{F}_{2^m}$, say, in two different ways: First of all, as before, by $\mathbb{F}_2[X]/(f)$ with f irreducible of degree m. Second, as residue class field of a place in a function field over $\mathbb{F}_2$, given by a $\mathcal{C}_{a,b}$ curve $\mathcal{C} : Y^a - F(X, Y) = 0$ with $b \approx a$. Suppose that the ideal (f) is totally split in $\mathbb{F}_2(\mathcal{C})$, and let $\mathfrak{f} = (f(X), Y - t(X))$ be an ideal of $\mathbb{F}_2[\mathcal{C}]$ above (f). Then the two homomorphisms with domain $\mathbb{F}_2[X, Y]$, given on the one hand by the reduction $\psi : \mathbb{F}_2[X, Y] \to \mathbb{F}_2[\mathcal{C}]$ modulo the curve equation and on the other hand by the evaluation map $\varphi : \mathbb{F}_2[X, Y] \to \mathbb{F}_2[X]$, $Y \mapsto t(X)$, are compatible with the reductions modulo $\mathfrak{f}$ and f:



By drawing a random polynomial $w \in \mathbb{F}_2[X, Y]$, one thus obtains a relation in $\mathbb{F}_{2^m}$ whenever both images under ψ and φ are smooth. Some technical complications stem from the fact that $\mathbb{F}_2[\mathcal{C}]$ is in general not a principal domain, so that instead of decomposing $\psi(w)$, one is limited to decomposing the principal ideal it generates. Apart from this, decomposition on the function field side amounts to factoring the norm of $\psi(w)$ and is thus reduced again to factoring univariate polynomials.

The degree a of the curve yields an additional degree of freedom; by choosing carefully the parameters, the degrees of the norm of $\psi(w)$ as well as of $\varphi(w)$ may

be bounded by $\log L_{2^m}(2/3)$. At first sight, the situation is unfavourable since two smoothness conditions have to be satisfied simultaneously instead of only one. But by (1), this influences only the constant, while the lower degree of the polynomials to be factored acts on the more important first parameter of the subexponential function. The worse constant, however, implies that the algorithms in $L(1/3)$ are not immediately faster than those in $L(1/2)$, but only from a certain input size on. The complexity of $L(1/3)$ is only heuristic since it relies on the assumption that the norm of a polynomial in $\mathbb{F}_2[\mathcal{C}]$ has the same smoothness probability as a random univariate polynomial of the same degree.

3.2. An algorithm for a special class of curves. It is now a natural question to ask whether it is possible to reach a complexity of $L(1/3)$ also for the discrete logarithm problem in Jacobians of curves over finite fields. Given the analogies between finite fields and Jacobians used in Section 2.5 to develop a unified theory for algorithms of complexity in $L(1/2)$, one might nourish some hope to similarly generalise the algorithms in $L(1/3)$ to curves. But the second way of representing $\mathbb{F}_{2^m}$ as a residue field in a function field (respectively, $\mathbb{F}_p$ as a residue field in a number field) does not seem to be parallelled in Jacobians. It is apparently impossible, for instance, to stack a second curve on top of the first one.

The solution to this problem presented in [**27**] turns this apparent obstacle into an advantage: Indeed, it is suggested to work *directly* with the curves that appear in the function field sieve. The algorithm is not limited to $\mathcal{C}_{a,b}$ curves. Let a_0 and b_0 be arbitrary positive constants. Consider a family of absolutely irreducible curves of genus g over a finite field $\mathbb{F}_q$ of the form

$$\mathcal{C} : Y^a + F(X, Y)$$

with $F(X, Y) \in \mathbb{F}_q[X, Y]$ of degree b in X and at most $a - 1$ in Y, where a and b are bounded by

$$a < a_0 g^{1/3} \mathcal{M}^{-1/3} \text{ and } b < b_0 g^{2/3} \mathcal{M}^{1/3} \tag{4}$$

with $\mathcal{M} = \frac{\log(g \log q)}{\log q} = \log_q(g \log q)$. To apply the smoothness result of Theorem 3, one furthermore has to impose that $g \geqslant (\log q)^\delta$ for some $\delta > 2$.

For instance, one may choose $a_0 > 0$ arbitrarily, fix $b_0 = \frac{2}{a_0}$ and consider $\mathcal{C}_{a,b}$ curves satisfying (4); this ensures that we are not speaking about the empty set.

Relations are created in the same way as in Adleman–DeMarrais–Huang's algorithm of Section 2.4: As principal divisors are zero in the Jacobian, it suffices to draw random polynomials $w = r(X) + s(X)Y$ and to verify whether their divisors are smooth; again, this amounts to factoring the norm of w in $\mathbb{F}_q[X]$.

Choosing as factor base the $L_{q^g}(1/3, d)$ smallest prime divisors and the degrees of r and s as $cg^{1/3}\mathcal{M}^{2/3}$, the following two properties hold:

- First of all, the smoothness probability of the norm is (heuristically) given by $1/L(1/3, e/d + o(1))$ with $e = (a_0 c + b_0)/3$.
- Second, the *sieving space*, that is, the set from which the tuple (r, s) is drawn, is sufficiently large. In fact, it would be possible to increase the smoothness probability by selecting r and s of even smaller (in the extreme case, constant) degrees; but then the number of choices for w would be so restricted that one would not even obtain a single relation on average. As in other subexponential algorithms, one has to ensure that the number of random choices at one's disposition is at least as large as the number of

smoothness tests carried out. This is the main obstacle to decreasing the complexity below $L(1/3)$.

By computing the Smith normal form of the relation matrix, one obtains the order and the structure of the Jacobian as a product of cyclic groups. With the optimal choice of the free parameters c and d, the complexity becomes

$$L_{q^g}\left(1/3, \frac{4}{3}\sqrt{a_0 c + b_0} + o(1)\right),$$

where c is the positive solution of the quadratic equation $c^2 - \frac{4}{9}a_0 c - \frac{4}{9}b_0 = 0$.

It remains to be seen how to compute discrete logarithms. One needs (as in the third stage of Algorithm 2) an additional relation containing the divisor Q whose logarithm is sought. Unfortunately, the size of Q cannot be controlled, and chances are that it is of order $\log L(1)$ rather than $\log L(2/3)$. Perturbing Q randomly by elements of the factor base, the smoothness theorem 3 implies that in average time $L(1/3)$, one may obtain a relation containing Q and prime divisors Q_i of size $\log L(2/3)$. It is possible to use an approach called *special q descent* in the context of factorisation, creating for each Q_i a relation containing it by considering functions $w = r(X) + s(X)Y$ passing through Q_i. But in order to have a reasonable chance of finding a relation, one needs to arrange some freedom for the degrees of r and s; with the additional restriction of passing through one of the Q_i, one again has to decompose a divisor of degree $\log L(1)$, and the process turns in circles.

The solution suggested in [**27**] consists in relaxing slightly the constraint on the running time. Let thus $\varepsilon > 0$ be fixed. In time $L(1/3+\varepsilon)$, one may create a relation containing Q and further prime divisors Q_i of degree $\log L(2/3 - \varepsilon)$. For each Q_i, a special q descent allows to replace it in time $L(1/3 + \varepsilon)$ by a linear combination of prime divisors $Q_{i,j}$ of degree $\log L(2/3 - 2\varepsilon)$; these $Q_{i,j}$ are again treated by a special q descent, and so forth. Whenever the degree of a $Q_{i,j,\ldots}$ drops below the barrier of $L(1/3 + \varepsilon)$, the descent returns primes of degree $\log L(1/3)$, which are elements of the factor base, and the process terminates.

This descent approach creates a tree in which all nodes have a degree in $O(g)$, whose height is bounded by $1/(3\varepsilon)$, and whose leaves are in the factor base. As ε is a constant, the number of nodes in the tree is polynomial in g and thus generously covered by any subexponential function. So the following result holds:

THEOREM 4 (heuristic). *Let there be given a family of curves $\mathcal{C}$ as above, satisfying in particular (4) and $g \geqslant (\log q)^\delta$ for some $\delta > 2$, and let $\varepsilon > 0$. Assuming heuristically that the divisors encountered during the algorithm have the same smoothness probability of Theorem 3 as random divisors of the same degree, discrete logarithms in the Jacobian of $\mathcal{C}$ can be computed in time $L_{q^g}(1/3 + \varepsilon, o(1))$.*

Concerning the constant of the subexponential complexity, it suffices to note that the existence of an algorithm in $L(1/3 + \varepsilon/2, c)$ for some constant c allows to reach $L(1/3 + \varepsilon, o(1))$ by (1).

The degrees a in X and b in Y of the curve may be balanced differently. Letting $a \approx g^\alpha$ and $b \approx g^{1-\alpha}$ for some α between $1/3$ and $1/2$, the algorithm for computing the group structure remains of complexity $L(1/3)$ (with a different constant depending on α), while the time for computing discrete logarithms becomes $L(\alpha + \varepsilon)$. When α drops below $1/3$, also the group structure computation becomes slower than $L(1/3)$; its complexity turns out to be $L(x(\alpha))$ for $x(\alpha) \in [1/3, 1/2]$

and, in particular, $L(1/2)$ for hyperelliptic curves. This is apparently the first natural occurrence of an algorithm with a subexponential complexity in which the first parameter is different from $1/3$ and $1/2$.

3.3. The low genus case. In the spirit of Section 2.6, Diem has considered in [**20**] a particular class of low genus curves, in which discrete logarithms are easier to compute. His ideas are independent of the algorithm of complexity $L(1/3)$ presented in the previous section, but it turns out that the gain with respect to general curves is in both cases due to a curve degree that is comparatively small for a given genus. Let again be given a family of curves with q tending to infinity, this time represented by plane models of fixed degree d instead of a fixed genus. The factor base is formed, as in Section 2.6, by the prime divisors of degree 1, otherwise said, the rational points on the curve. Relations are created, as in the algorithm of Section 3.2 and as in Adleman–DeMarrais–Huang's algorithm, by computing divisors of polynomials; so like these two algorithms, Diem's approach is heuristic. He considers furthermore only the simplest polynomials, namely lines. By Bézout's theorem, they intersect the curve in d points (that may have coordinates in an extension field); so a relation is obtained whenever a polynomial of degree d factors into linear factors over the base field, as opposed to Section 2.6, in which the polynomial was of degree g. Using the double prime variation, one obtains an algorithm of heuristic complexity $O(q^{2-2/d})$, measured in arithmetic operations. Diem suggests an additional trick to lower the complexity; he restricts to lines drawn between two points that are already in the factor base. Then a polynomial of degree only $d-2$ has to split into linear factors to yield a relation, and the complexity of the discrete logarithm algorithm drops to $O(q^{2-2/(d-2)})$. This algorithm is preferable to the one described in Section 2.6 whenever $d-2 < g$. Hyperelliptic curves are not concerned, but the impact on $\mathcal{C}_{a,b}$ curves with $3 \leqslant a < b$ is dramatic. The equations $d = b$ and $g = \frac{(a-1)(b-1)}{2}$ imply that discrete logarithms are obtained with $O(q^{2-2(a-1)/(2g-(a-1))})$ operations. In particular, in the case $a = 3$ and $b = 4$ the complexity is $O(q)$; so the discrete logarithm problem in non-hyperelliptic $\mathcal{C}_{a,b}$ curves of genus 3 is not harder than in hyperelliptic curves of genus 2 *defined over the same finite field*, while the bit length of the group order is 50 % higher and the arithmetic is considerably more involved. This result implies that non-hyperelliptic curves are not suited for the implementation of discrete logarithm based cryptosystems.

4. Implementations

The latest data points for computing discrete logarithms with a generic algorithm are from 2002 and 2004 and concern elliptic curves over prime fields and fields of characteristic 2 of 109 bits [**13, 14**]; the 2004 computation involved 2600 processors running over 17 months.

A subexponential algorithm for hyperelliptic curves has first been implemented by Flassenberg and Paulus [**29**]. Their largest example, a curve of genus 12 over $\mathbb{F}_{11}$, is far from reaching a cryptographic parameter size; since the cardinalities of these high genus curves were unknown, the authors had to resort to expensive Hermite normal form computation instead of solving a sparse linear system. Gaudry reports on an implementation of the algorithm of Section 2.6 (without large primes) in [**36**]; his largest examples, curves of genus 6 over $\mathbb{F}_{5026243}$ respectively $\mathbb{F}_{2^{23}}$, surpass the

generic record of the previous paragraph and are very close to cryptographic group orders.

The algorithm of Section 3.3, including the double large prime variation, has been implemented by Diem and Thomé for a $\mathcal{C}_{3,4}$ curve of genus 3 over $\mathbb{F}_{2^{31}}$, see [**21**]. Their computation taking only a few days with the relation collection carried out on a single CPU, this should rather be seen as a proof of concept for the algorithm than as a benchmark on what is achievable today. The authors estimate that discrete logarithms on a $\mathcal{C}_{3,4}$ curve with a group order of 111 bits could be obtained by an effort comparable to that of factoring a 664 bit RSA integer.

5. Future research

The algorithm of Section 3.2 of complexity $L(1/3 + \varepsilon)$ for computing discrete logarithms in certain curves opens a new direction of research. During the 10th Workshop on Elliptic Curve Cryptography (ECC 2006), Diem has announced an algorithm of complexity $L(1/3)$ inspired by these ideas, but with a quite different point of view [**19**]; for the time being, it is unclear whether his class of curves is different from the one considered in Section 3.2. It would be interesting to obtain a complete classification of the curves that are subject to a subexponential attack of complexity better than $L(1/2)$.

In a recent preprint [**67**], Smith has found a novel attack on certain hyperelliptic curves of genus 3. He explicitly computes an isogeny to a non-hyperelliptic curve of genus 3, which allows to transport the discrete logarithm problem and to solve it via the algorithm of Section 3.3. Heuristically, the attack applies to about one out of five hyperelliptic curves of genus 3. However, by considering more general isogenies, it appears likely that the result could be extended to other curves, which would cast further doubt on the use of genus 3 curves in cryptography.

References

1. Fatima K. Abu Salem and Kamal Khuri-Makdisi, *Fast Jacobian group operations for $C_{3,4}$ curves over a large finite field*, LMS Journal of Computation and Mathematics **10** (2007), 307–328.
2. Leonard M. Adleman, *The function field sieve*, Algorithmic Number Theory (Berlin) (Leonard M. Adleman and Ming-Deh Huang, eds.), Lecture Notes in Computer Science, vol. 877, Springer-Verlag, 1994, pp. 108–121.
3. Leonard M. Adleman, Jonathan DeMarrais, and Ming-Deh Huang, *A subexponential algorithm for discrete logarithms over the rational subgroup of the Jacobians of large genus hyperelliptic curves over finite fields*, Algorithmic Number Theory (Berlin) (Leonard M. Adleman and Ming-Deh Huang, eds.), Lecture Notes in Computer Science, vol. 877, Springer-Verlag, 1994, pp. 28–40.
4. Seigo Arita, *Algorithms for computations in Jacobian group of C_{ab} curve and their application to discrete-log based public key cryptosystems*, IEICE Transactions **J82-A** (1999), no. 8, 1291–1299, In Japanese. English translation in the proceedings of the Conference on The Mathematics of Public Key Cryptography, Toronto 1999, or [**6**].
5. ———, *An addition algorithm in Jacobian of C_{34} curve*, Information Security and Privacy — ACISP 2003 (Berlin) (Rei Safavi-Naini and Jennifer Seberry, eds.), Lecture Notes in Computer Science, vol. 2727, Springer-Verlag, 2003, pp. 93–105.
6. ———, *An addition algorithm in Jacobian of C_{ab} curves*, Discrete Applied Mathematics **130** (2003), no. 1, 13–31.
7. Abdolali Basiri, Andreas Enge, Jean-Charles Faugère, and Nicolas Gürel, *Implementing the arithmetic of $C_{3,4}$ curves*, Algorithmic Number Theory — ANTS-VI (Berlin) (Duncan Buell, ed.), Lecture Notes in Computer Science, vol. 3076, Springer-Verlag, 2004, pp. 87–101.

8. ———, *The arithmetic of Jacobian groups of superelliptic cubics*, Mathematics of Computation **74** (2005), no. 249, 389–410.
9. Renet Lovorn Bender and Carl Pomerance, *Rigorous discrete logarithm computations in finite fields via smooth polynomials*, Computational Perspectives on Number Theory: Proceedings of a Conference in Honor of A.O.L. Atkin (D. A. Buell and J. T. Teitelbaum, eds.), Studies in Advanced Mathematics, vol. 7, American Mathematical Society, 1998, pp. 221–232.
10. Ian Blake, Gadiel Seroussy, and Nigel Smart, *Elliptic curves in cryptography*, London Mathematical Society Lecture Note Series, vol. 265, Cambridge University Press, Cambridge, 1999.
11. N. G. de Bruijn, *The asymptotic behaviour of a function occurring in the theory of primes*, J. Indian Math. Soc. **15** (1951), 15–32.
12. David G. Cantor, *Computing in the Jacobian of a hyperelliptic curve*, Mathematics of Computation **48** (1987), no. 177, 95–101.
13. Certicom, *Certicom announces elliptic curve cryptosystem (ECC) challenge winner*, 2002, `http://www.certicom.com/index.php?action=company,press_archive&view=121`.
14. ———, *Certicom announces elliptic curve cryptography challenge winner*, 2004, `http://www.certicom.com/index.php?action=company,press_archive&view=307`.
15. Don Coppersmith, *Fast evaluation of logarithms in fields of characteristic two*, IEEE Transactions on Information Theory **30** (1984), no. 4, 587–594.
16. Jean-Marc Couveignes, *Algebraic groups and discrete logarithm*, Public-Key Cryptography and Computational Number Theory (Berlin) (K. Alster, J. Urbanowicz, and H. C. Williams, eds.), De Gruyter, 2001, pp. 17–27.
17. Diderot and d'Alembert (eds.), *Encyclopédie, ou dictionnaire raisonné des sciences, des arts et des métiers, par une société de gens de lettres*, Briasson, David, Le Breton et Durand, 1751–1772.
18. Claus Diem, *The GHS attack in odd characteristic*, Journal of the Ramanujan Mathematical Society **18** (2003), no. 1, 1–32.
19. ———, *An index calculus algorithm for non-singular plane curves of high genus*, 2006, Slides, 10th Workshop on Elliptic Curve Cryptography, Toronto, September 18–20, `http://www.cacr.math.uwaterloo.ca/conferences/2006/ecc2006/diem.pdf`.
20. ———, *An index calculus algorithm for plane curves of small degree*, Algorithmic Number Theory — ANTS-VII (Berlin) (Florian Hess, Sebastian Pauli, and Michael Pohst, eds.), Lecture Notes in Computer Science, vol. 4076, Springer-Verlag, 2006, pp. 543–557.
21. Claus Diem and Emmanuel Thomé, *Index calculus in class groups of non-hyperelliptic curves of genus three*, HAL-INRIA 107290, INRIA, 2007, http://hal.inria.fr/inria-00107290/, to appear in *Journal of Cryptology*.
22. P. G. Lejeune Dirichlet and R. Dedekind, *Vorlesungen über Zahlentheorie*, 2nd ed., Vieweg, Braunschweig, 1871.
23. Andreas Enge, *The extended Euclidian algorithm on polynomials, and the computational efficiency of hyperelliptic cryptosystems*, Designs, Codes and Cryptography **23** (2001), no. 1, 53–74.
24. ———, *A general framework for subexponential discrete logarithm algorithms in groups of unknown order*, Finite Geometries (Dordrecht) (A. Blokhuis, J. W. P. Hirschfeld, D. Jungnickel, and J. A. Thas, eds.), Developments in Mathematics, vol. 3, Kluwer Academic Publishers, 2001, pp. 133–146.
25. ———, *Computing discrete logarithms in high-genus hyperelliptic Jacobians in provably subexponential time*, Mathematics of Computation **71** (2002), no. 238, 729–742.
26. Andreas Enge and Pierrick Gaudry, *A general framework for subexponential discrete logarithm algorithms*, Acta Arithmetica **102** (2002), no. 1, 83–103.
27. ———, *An $L(1/3+\varepsilon)$ algorithm for the discrete logarithm problem for low degree curves*, Advances in Cryptology — Eurocrypt 2007 (Berlin) (Moni Naor, ed.), Lecture Notes in Computer Science, vol. 4515, Springer-Verlag, 2007, pp. 367–382.
28. Andreas Enge and Andreas Stein, *Smooth ideals in hyperelliptic function fields*, Mathematics of Computation **71** (2002), no. 239, 1219–1230.
29. Ralf Flassenberg and Sachar Paulus, *Sieving in function fields*, Experimental Mathematics **8** (1999), no. 4, 339–349.
30. Stéphane Flon and Roger Oyono, *Fast arithmetic on Jacobians of Picard curves*, Public Key Cryptography — PKC 2004 (Berlin) (Feng Bao, Robert Deng, and Jianying Zhou, eds.), Lecture Notes in Computer Science, vol. 2947, Springer-Verlag, 2004, pp. 55–68.

31. Gerhard Frey, *Applications of arithmetical geometry to cryptographic constructions*, Finite Fields and Applications — Proceedings of The Fifth International Conference on Finite Fields and Applications F_q5, held at the University of Augsburg, Germany, August 2–6, 1999 (Berlin) (Dieter Jungnickel and Harald Niederreiter, eds.), Springer-Verlag, 2001, pp. 128–161.
32. Gerhard Frey and Hans-Georg Rück, *A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves*, Mathematics of Computation **62** (1994), no. 206, 865–874.
33. S. D. Galbraith, S. M. Paulus, and N. P. Smart, *Arithmetic on superelliptic curves*, Mathematics of Computation **71** (2002), no. 237, 393–405.
34. P. Gaudry, F. Hess, and N. P. Smart, *Constructive and destructive facets of Weil descent on elliptic curves*, Journal of Cryptology **15** (2002), 19–46.
35. P. Gaudry, E. Thomé, N. Thériault, and C. Diem, *A double large prime variation for small genus hyperelliptic index calculus*, Mathematics of Computation **76** (2007), no. 257, 475–492.
36. Pierrick Gaudry, *An algorithm for solving the discrete log problem on hyperelliptic curves*, Advances in Cryptology — EUROCRYPT 2000 (Berlin) (Bart Preneel, ed.), Lecture Notes in Computer Science, vol. 1807, Springer-Verlag, 2000, pp. 19–34.
37. ———, *Hyperelliptic curves and the HCDLP*, Advances in Elliptic Curve Cryptography (Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart, eds.), Cambridge University Press, Cambridge, 2005, pp. 133–150.
38. ———, *Fast genus 2 arithmetic based on theta functions*, Journal of Mathematical Cryptology **1** (2007), no. 3, 243–265.
39. Daniel M. Gordon, *Discrete logarithms in GF(p) using the number field sieve*, SIAM Journal on Discrete Mathematics **6** (1993), no. 1, 124–138.
40. Florian Heß, *Computing Riemann–Roch spaces in algebraic function fields and related topics*, Journal of Symbolic Computation **33** (2002), no. 4, 425–445.
41. Florian Hess, *Computing relations in divisor class groups of algebraic curves over finite fields*, Preprint, `http://www.math.tu-berlin.de/~hess/personal/dlog.ps.gz`, 2004.
42. ———, *Weil descent attacks*, Advances in Elliptic Curve Cryptography (Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart, eds.), Cambridge University Press, Cambridge, 2005, pp. 151–180.
43. Antoine Joux, Reynald Lercier, Nigel Smart, and Frederik Vercauteren, *The number field sieve in the medium prime case*, Advances in Cryptology — CRYPTO 2006 (Berlin) (Cynthia Dwork, ed.), Lecture Notes in Computer Science, vol. 4117, Springer-Verlag, 2006, pp. 326–344.
44. Kamal Khuri-Makdisi, *Linear algebra algorithms for divisors on an algebraic curve*, Mathematics of Computation **73** (2004), no. 245, 333–357.
45. John Knopfmacher, *Abstract analytic number theory*, North-Holland Mathematical Library, vol. 12, North-Holland Publishing Company, Amsterdam, 1975.
46. M. Kraïtchik, *Théorie des nombres*, Gauthier-Villars, Paris, 1922.
47. Cornelius Lanczos, *Solution of systems of linear equations by mimimized iterations*, Journal of Research of the National Bureau of Standards **49** (1952), no. 1, 33–53.
48. Tanja Lange, *Formulae for arithmetic on genus 2 hyperelliptic curves*, Applicable Algebra in Engineering, Communication and Computing **15** (2005), no. 5, 295–328.
49. E. Manstavičius, *Remarks on the semigroup elements free of large prime factors*, Lithuanian Mathematical Journal **32** (1992), no. 4, 400–409.
50. ———, *Semigroup elements free of large prime factors*, New Trends in Probability and Statistic (F. Schweiger and E. Manstavičius, eds.), 1992, pp. 135–153.
51. Daniel A. Marcus, *Number fields*, Springer-Verlag, New York, 1977.
52. Ueli M. Maurer and Stefan Wolf, *The relationship between breaking the Diffie–Hellman protocol and computing discrete logarithms*, SIAM Journal on Computing **28** (1999), no. 5, 1689–1721.
53. Alfred J. Menezes, Tatsuaki Okamoto, and Scott A. Vanstone, *Reducing elliptic curve logarithms to logarithms in a finite field*, IEEE Transactions on Information Theory **39** (1993), no. 5, 1639–1646.
54. Volker Müller, Andreas Stein, and Christoph Thiel, *Computing discrete logarithms in real quadratic congruence function fields of large genus*, Mathematics of Computation **68** (1999), no. 226, 807–822.

55. Koh-ichi Nagao, *Index calculus attack for Jacobian of hyperelliptic curves of small genus using two large primes*, Japan Journal of Industrial and Applied Mathematics **24** (2007), no. 3, 289–305.
56. V. I. Nechaev, *Complexity of a determinate algorithm for the discrete logarithm*, Mathematical Notes **55** (1994), no. 2, 165–172.
57. P. C. van Oorschot and M. J. Wiener, *Parallel collision search with cryptanalytic applications*, Journal of Cryptology **12** (1999), no. 1, 1–28.
58. Stephen C. Pohlig and Martin E. Hellman, *An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance*, IEEE Transactions on Information Theory **24** (1978), no. 1, 106–110.
59. J. M. Pollard, *Monte Carlo methods for index computation* (mod p), Mathematics of Computation **32** (1978), no. 143, 918–924.
60. Carl Pomerance, *Fast, rigorous factorization and discrete logarithm algorithms*, Discrete Algorithms and Complexity, Proceedings of the Japan–US Joint Seminar, June 4–6, 1986, Kyoto, Japan (Orlando) (David S. Johnson, Takao Nishizeki, Akihiro Nozaki, and Herbert S. Wolf, eds.), Perspectives in Computing, vol. 15, Academic Press, 1987, pp. 119–143.
61. Takakazu Satoh and Kiyomichi Araki, *Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves*, Commentarii Mathematici Universitatis Sancti Pauli **47** (1998), no. 1, 81–92, Errata in vol. 48 (2):211–213, 1999.
62. I. A. Semaev, *Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p*, Mathematics of Computation **67** (1998), no. 221, 353–356.
63. Martin Seysen, *A probabilistic factorization algorithm with quadratic forms of negative discriminant*, Mathematics of Computation **48** (1987), no. 178, 757–780.
64. Daniel Shanks, *The infrastructure of a real quadratic number field and its applications*, Proc. 1972 Number Th. Conf. (Boulder (Colorado)), 1972, pp. 217–224.
65. Victor Shoup, *Lower bounds for discrete logarithms and related problems*, Advances in Cryptology — EUROCRYPT '97 (Berlin) (Walter Fumy, ed.), Lecture Notes in Computer Science, vol. 1233, Springer-Verlag, 1997, pp. 256–266.
66. Nigel P. Smart, *The discrete logarithm problem on elliptic curves of trace one*, Journal of Cryptology **12** (1999), no. 3, 193–196.
67. Benjamin Smith, *Isogenies and the discrete logarithm problem on Jacobians of genus 3 hyperelliptic curves*, Preprint, 2007.
68. Edlyn Teske, *On random walks for Pollard's rho method*, Mathematics of Computation **70** (2001), no. 234, 809–825.
69. Nicolas Thériault, *Index calculus attack for hyperelliptic curves of small genus*, Advances in Cryptology — ASIACRYPT 2003 (Berlin) (Chi Sung Laih, ed.), Lecture Notes in Computer Science, vol. 2894, Springer-Verlag, 2003, pp. 75–92.
70. André Weil, *Courbes algébriques et variétés abéliennes*, Hermann, Paris, 1971.
71. Douglas H. Wiedemann, *Solving sparse linear equations over finite fields*, IEEE Transactions on Information Theory **32** (1986), no. 1, 54–62.

INRIA Futurs & Laboratoire d'Informatique (CNRS/UMR 7161), École polytechnique, 91128 Palaiseau Cedex, France

E-mail address: enge@lix.polytechnique.fr