MATHEMATICS OF COMPUTATION
Volume 80, Number 273, January 2011, Pages 443–475
S 0025-5718(2010)02281-1
Article electronically published on August 23, 2010

ON THE DISCRETE LOGARITHM PROBLEM IN CLASS GROUPS OF CURVES

CLAUS DIEM

Abstract. We study the discrete logarithm problem in degree 0 class groups of curves over finite fields, with particular emphasis on curves of small genus. We prove that for every fixed $g \geq 2$, the discrete logarithm problem in degree 0 class groups of curves of genus g can be solved in an expected time of $\tilde{\mathcal{O}}(q^{2-\frac{2}{g}})$, where $\mathbb{F}_q$ is the ground field. This result generalizes a corresponding result for hyperelliptic curves given in imaginary quadratic representation with cyclic degree 0 class group, and just as this previous result, it is obtained via an index calculus algorithm with double large prime variation.

Generalizing this result, we prove that for fixed $g_0 \geq 2$ the discrete logarithm problem in class groups of all curves $\mathcal{C}/\mathbb{F}_q$ of genus $g \geq g_0$ can be solved in an expected time of $\tilde{\mathcal{O}}((q^g)^{\frac{2}{g_0}(1-\frac{1}{g_0})})$ and in an expected time of $\tilde{\mathcal{O}}(\#\mathrm{Cl}^0(\mathcal{C})^{\frac{2}{g_0}(1-\frac{1}{g_0})})$.

As a complementary result we prove that for any fixed $n \in \mathbb{N}$ with $n \geq 2$ the discrete logarithm problem in the groups of rational points of elliptic curves over finite fields $\mathbb{F}_{q^n}$, q a prime power, can be solved in an expected time of $\tilde{\mathcal{O}}(q^{2-\frac{2}{n}})$.

Furthermore, we give an algorithm for the efficient construction of a uniformly randomly distributed effective divisor of a specific degree, given the curve and its L-polynomial.

1. Introduction

With a deterministic algorithm, one can solve the discrete logarithm problem in degree 0 class groups of curves $\mathcal{C}/\mathbb{F}_q$ in a time of $\tilde{\mathcal{O}}(\#\,\mathrm{Cl}^0(\mathcal{C})^{\frac{1}{2}})$.[1]

This work is motivated by the following question: Under which conditions on the genera of the curves can one solve the discrete logarithm problem in degree 0 class groups of curves in an expected time of $o(\#\,\mathrm{Cl}^0(\mathcal{C})^{\frac{1}{2}})$? The underlying model of computation is here and throughout this work a randomized random access machine model with commands as in [AHU74] and an additional command to choose 0 or 1 uniformly at random. The running time is then determined with the logarithmic cost function.

We show that the condition that the genus be ≥ 3 is sufficient. Indeed, we prove that under this condition the discrete logarithm problem can be solved in an

Received by the editor November 26, 2008 and, in revised form, March 25, 2009.
2010 *Mathematics Subject Classification.* Primary 11Y16; Secondary 14G50, 68Q25, 94A60.
Key words and phrases. Discrete logarithm problem, curves, index calculus.

[1] This result follows with the baby-step-giant-step algorithm combined with a precomputation. It is based on a particular representation of the input curve and input divisor classes which is described at the end of Section 2.

expected time of $\tilde{\mathcal{O}}(\#\operatorname{Cl}^0(\mathcal{C})^{\frac{4}{9}})$. It is also natural to bound running times in terms of g^g, where g is the genus. Again we prove that under the condition that the genus be ≥ 3 one can obtain an expected running time of $\tilde{\mathcal{O}}((q^g)^{\frac{4}{9}})$.

Our main contribution is the following theorem.

Theorem 1. *Let some natural number $g \geq 2$ be fixed. Then the discrete logarithm problem in the degree 0 class groups of curves of genus g over finite fields can be solved in an expected time of*

$$\tilde{\mathcal{O}}(q^{2-\frac{2}{g}}),$$

where $\mathbb{F}_q$ is the ground field of the curve.

The algorithms for this theorem have storage requirements of $\tilde{\mathcal{O}}(q^{1-\frac{1}{g}+\frac{1}{g^2}})$. More concretely, although the algorithms are randomized, there exists a function in $\tilde{\mathcal{O}}(q^{1-\frac{1}{g}+\frac{1}{g^2}})$ such that the storage requirements are bounded by this function for every run.

This theorem is obtained via index calculus algorithms with double large prime variation. A corresponding result for hyperelliptic curves in imaginary quadratic representation and with cyclic degree 0 class groups was obtained in [GTTD07], and *on a heuristic basis* an obvious generalization of the algorithm for the theorem in [GTTD07] already gives rise to the result in Theorem 1. A related work is [Nag07], in which a similar algorithm is given. We note, however, that the main "theorem" in [Nag07] is not proven but only established on a heuristic basis.

The algorithms in both works are also based on the index calculus method, and both follow the "double large prime variation strategy". The algorithm presented in this work is quite closely related to—albeit different from—the algorithm in [Nag07].

From Theorem 1 and results in [Heß05], the following theorem follows quite easily.

Theorem 2. *Let some natural number $g_0 \geq 2$ be fixed. Then the discrete logarithm problem in the degree 0 class groups of curves $\mathcal{C}/\mathbb{F}_q$ of genus $g \geq g_0$ can be solved in an expected time of*

$$\tilde{\mathcal{O}}((\#\operatorname{Cl}^0(\mathcal{C}))^{\frac{2}{g_0}(1-\frac{1}{g_0})})$$

as well as in an expected time of

$$\tilde{\mathcal{O}}((q^g)^{\frac{2}{g_0}(1-\frac{1}{g_0})}).$$

As a complementary result to Theorem 1, we prove the following theorem on the elliptic curve discrete logarithm problem over extension fields.

Theorem 3. *Let some natural number $n \geq 2$ be fixed. Then the discrete logarithm problem in the groups of rational points of elliptic curves over finite fields $\mathbb{F}_{q^n}$, q being a prime power, be solved in an expected time of*

$$\tilde{\mathcal{O}}(q^{2-\frac{2}{n}}).$$

The algorithms for this theorem are similar to the one for Theorem 1. The key difference is that we now generate relations by solving systems of polynomial equations over $\mathbb{F}_q$, in contrast to factoring divisors.

Again, *on a heuristic basis*, the result in Theorem 3 was already established in [Gau09] with a similar algorithm.

Overview. We now give an overview on the rest of this work. In the next section we discuss how we represent curves, divisors and divisor classes. At the end of the section we indicate how the input data to the algorithms are represented, and we discuss the representation of divisor classes by unique bit strings. Note that unique representation of group elements is crucial for the baby-step-giant-step algorithm and thus for the result mentioned at the beginning of this work. In Section 3 we give the algorithm for Theorem 1. For this we first give an algorithm which operates well if applied to an instance as in Theorem 1 together with a generating system, and we give an algorithm which with a high probability outputs such a generating system. We show how these two algorithms can be combined in such a way that Theorem 1 follows. Building on Theorem 1, in Section 4 we give an algorithm for Theorem 2. In Section 5 we show how the algorithm for Theorem 1 can be modified in such a way that one obtains Theorem 3. This modification as well as the analysis of the algorithm rely crucially on algorithms and results in [Die10].

2. Representations and basic computations

In this section we describe how we represent the basic objects of our computation via bit strings: Curves (always assumed to be smooth, proper and geometrically irreducible) over finite fields and divisors and divisor classes on such curves. We also discuss basic computations with these objects. We fix some notation as we go along.

For the representation of divisors we follow an ideal and function field theoretic approach which is inspired by the number field theoretic analog. This approach is convenient for several reasons. First, one can easily adapt the usual algorithms for arithmetic with ideals in number fields (as for example in [Coh96]) to the function field theoretic setting. Second, using the ideal theoretic approach an efficient and simple algorithm for the computation of Riemann-Roch spaces $L(D)$ can be given ([Heß01]). The following exposition is heavily inspired by [Heß01]. Further information, including proofs of all claims, can be found in [Die08, Chapter 2].

Let k be a field. Then we define $\mathbb{A}^1_k := \mathrm{Spec}(k[\mathrm{x}])$ and $\mathbb{P}^1_k := \mathrm{Proj}(k[\mathrm{X}, \mathrm{Y}])$; we identify $k[\mathrm{x}]$ with $k[\frac{\mathrm{X}}{\mathrm{Y}}] \subset k(\mathbb{P}^1)$ via $\mathrm{x} \longleftrightarrow \frac{\mathrm{X}}{\mathrm{Y}}$, and we define $\infty := (0:1) \in \mathbb{P}^1(k)$. We obtain a canonical inclusion $\mathbb{A}^1_k \hookrightarrow \mathbb{P}^1_k$ and with this inclusion a decomposition $\mathbb{P}^1(\overline{k}) = \mathbb{A}^1(\overline{k}) \,\dot{\cup}\, \{\infty\}$. Furthermore, we set $\mathbb{P}^2_k := \mathrm{Proj}(k[X, Y, Z])$.

By a *curve* over k we mean a smooth, proper and geometrically irreducible 1-dimensional k-scheme.

Curves. In the following, q is always the power of a prime p, and we set $k := \mathbb{F}_q$.

We represent curves via *plane models*: Let $\mathcal{C}/k$ be a curve. Then by a *plane model* of $\mathcal{C}$ we mean a 1-dimensional closed subscheme $\mathcal{C}_{pm}$ of $\mathbb{P}^2_k$ which is birational to $\mathcal{C}$. We represent such a plane model $\mathcal{C}_{pm}$ by a defining homogeneous polynomial $F(X, Y, Z)$. Note that the degree of $\mathcal{C}_{pm}$ equals the degree of F; we denote this degree by d. Furthermore, we denote the genus of $\mathcal{C}$ by g.

The following proposition is [Heß01, Theorem 56].

Proposition 2.1. *Every curve over a finite field has a plane model of degree $\mathcal{O}(g)$ (uniformly over all finite fields).*

Let us now fix a curve $\mathcal{C}$ of genus ≥ 1 with a birational morphism to a plane model: $\pi : \mathcal{C} \longrightarrow \mathcal{C}_{pm}$.

We set $x := \frac{X}{Z}, y := \frac{Y}{Z}$ and $f(x,y) := \frac{F(X,Y,Z)}{Z^d}$, and we denote the induced functions on $\mathcal{C}$ also by x, y. We assume that the covering $x : \mathcal{C} \longrightarrow \mathbb{P}^1_k$ is separable (otherwise the covering y is separable and we can interchange x and y), and we set $r := \deg(x) = [k(\mathcal{C}) : k(\mathrm{x})]$.

Divisors. We now come to closed points and divisors. For this, we consider the subrings $(x_*\mathcal{O}_\mathcal{C})(\mathbb{A}^1_k)$ and $(x_*\mathcal{O}_\mathcal{C})_\infty$ of $k(\mathcal{C})$; these rings are the integral closures of $k[\mathrm{x}]$ and $k[\frac{1}{\mathrm{x}}]_{(\frac{1}{\mathrm{x}})}$ in $k(\mathcal{C})$. They are called *finite* and *infinite order* of $k(\mathcal{C})$ (with respect to x).

Now the closed points of $\mathcal{C}$ correspond to the places of $k(\mathcal{C})$, and these in turn correspond to maximal ideals of $(x_*\mathcal{O}_\mathcal{C})(\mathbb{A}^1_k)$ as well as $(x_*\mathcal{O}_\mathcal{C})_\infty$.

The main idea for the ideal representation of points is to fix bases of these orders and to represent the points of $\mathcal{C}$ by "nice" $k[\mathrm{x}]$-, respectively, $k[\frac{1}{\mathrm{x}}]_{(\frac{1}{\mathrm{x}})}$-bases of the corresponding maximal ideals. In order to have better control over the size of this representation, we consider, however, prime ideals of $(x_*\mathcal{O}_\mathcal{C})(\frac{1}{\mathrm{x}}(\mathbb{A}^1_k))$ with support "at infinity" instead of prime ideals in $(x_*\mathcal{O}_\mathcal{C})_\infty$.

Now divisors can be represented in two ways. The first way is to take the representation of points as a basis for what is called a *free representation*: The divisors are represented by their support and the coefficient vector, and the points are represented by maximal ideals of the two orders.

Another way to represent divisors is to consider the isomorphisms

$$\begin{aligned} \mathrm{Div}(\mathcal{C}) &\xrightarrow{\sim} \mathrm{Div}((x_*\mathcal{O}_\mathcal{C})(\mathbb{A}^1_k)) \times \mathrm{Div}((x_*\mathcal{O}_\mathcal{C})_\infty) \\ &\xrightarrow{\sim} I((x_*\mathcal{O}_\mathcal{C})(\mathbb{A}^1_k)) \times I((x_*\mathcal{O}_\mathcal{C})_\infty)\ , \end{aligned} \tag{1}$$

where the first isomorphism is induced by pullback and the second isomorphism is induced by the canonical isomorphisms between the divisor and ideal groups. Via these isomorphisms every divisor corresponds to a pair of fractional ideals, a "finite" and an "infinite" fractional ideal. To represent the "infinite" fractional ideal, one furthermore applies the canonical inclusion $I((x_*\mathcal{O}_\mathcal{C})_\infty) \hookrightarrow I((x_*\mathcal{O}_\mathcal{C})(\frac{1}{\mathrm{x}}(\mathbb{A}^1_k)))$. We call this representation of divisors the *joint ideal representation.*

Notation 2.2. Let D be a divisor on $\mathcal{C}$. Then the associated "finite" fractional ideal in $I((x_*\mathcal{O}_\mathcal{C})(\mathbb{A}^1_k))$ is denoted by $I^{\mathrm{fin}}(D)$. The associated "infinite" fractional ideal in $I((x_*\mathcal{O}_\mathcal{C})(\frac{1}{\mathrm{x}}(\mathbb{A}^1_k)))$ is denoted by $I^\infty(D)$, and the corresponding ideal in $I((x_*\mathcal{O}_\mathcal{C})_\infty)$ (the localization of $I^\infty(D)$ at ∞) by $(I^\infty(D))_\infty$.

Note here that $I^\infty(D)$ by definition only has support "at infinity", and $(I^\infty(D))_\infty$ is the ideal generated by $I^\infty(D)$ inside $(x_*\mathcal{O}_\mathcal{C})_\infty$.

We now need to describe how we represent the fractional ideals of the orders $(x_*\mathcal{O}_\mathcal{C})(\mathbb{A}^1_k)$ and $(x_*\mathcal{O}_\mathcal{C})(\frac{1}{\mathrm{x}}(\mathbb{A}^1_k))$.

Let $f(x,y) = \sum_{i=0}^r a_i(x)\, y^i$. Let $\tilde{a}_i(\mathrm{x}) := a_i(\mathrm{x})\, a_r^{r-i-1}(\mathrm{x})$ for $i = 0, \ldots, r$, $\tilde{y} := a_r(\mathrm{x})\, y \in k(\mathcal{C})$ and $\tilde{\tilde{y}} := \frac{\tilde{y}}{\mathrm{x}^c} \in k(\mathcal{C})$ with $c := \max\{\lceil \frac{\deg(\tilde{a}_i(\mathrm{x}))}{r-i} \rceil \mid 0 \le i \le r\}$. Then $\tilde{y}$ is integral over $k[\mathrm{x}]$ as it is a root of the monic polynomial $\sum_{i=0}^r \tilde{a}_i(\mathrm{x}) \cdot t^i \in k[\mathrm{x}][t]$ and $\tilde{\tilde{y}}$ is integral over $k[\frac{1}{\mathrm{x}}]$ as it is a root of the monic polynomial $\sum_{i=0}^r \frac{\tilde{a}_i(\mathrm{x})}{\mathrm{x}^{c(r-i)}} \cdot t^i \in k[\frac{1}{\mathrm{x}}][t]$.

Now let $w_1, \ldots, w_r$ be the HNF-basis (HNF = Hermite normal form) of $(x_*\mathcal{O}_\mathcal{C})(\mathbb{A}^1_k)$ with respect to $1, \tilde{y}, \ldots, \tilde{y}^{r-1}$. Then we represent a fractional ideal of $(x_*\mathcal{O}_\mathcal{C})(\mathbb{A}^1_k)$ by the coordinate vectors of its HNF-basis with respect to $w_1, \ldots, w_r$.

We proceed similarly with the "infinite order", substituting $\tilde{y}$ by $\tilde{\tilde{y}}$.

We remark that one can compute the HNF-basis of $(x_*\mathcal{O}_\mathcal{C})(\mathbb{A}^1_k)$ (deterministically) in a time which is polynomially bounded in $d \cdot \log(q)$, following ideas by Zassenhaus and by Grauert and Remmert.

The input to all algorithms for the following computations with divisors and divisor classes consists of a curve, represented by a plane model, together with HNF-bases for the two orders and some additional data.

Inspired by [Heß01] we define:

Definition 2.3. Let D be a divisor on $\mathcal{C}$, and let D_+ be the "positive part" and D_- the "negative part" of D (such that $D_+, D_- \geq 0$, D_+ and D_- have disjoint support and $D = D_+ - D_-$). Then the *height* of D is $\text{ht}(D) := \max\{\deg(D_+), \deg(D_-)\}$.

Now basic divisor arithmetic (addition and inversion) in both free as well as joint ideal representation can be performed in a time which is polynomially bounded in $d \cdot h \cdot \log(q)$, where h is the maximum of the heights of the divisors involved.

By this result it is obvious that given a divisor D in free ideal representation, one can compute (deterministically) the corresponding divisor in joint ideal representation in a time which is polynomially bounded in $d \cdot \text{ht}(D) \cdot \log(q)$. The opposite computation is possible with a randomized algorithm whose expected running time is polynomially bounded in $d \cdot \text{ht}(D) \cdot \log(q)$. The randomization is only required to factorize polynomials over finite fields. By using Berlekamp's absolute algorithm for this task, one then obtains a deterministic algorithm with a running time of $p \cdot (d \cdot \text{ht}(D) \cdot \log(q))^{\mathcal{O}(1)}$, where, as indicated above, p is the characteristic.

For arithmetic in degree 0 class groups the computation of Riemann-Roch spaces $L(D)$ for a divisor D is crucial. An efficient and simple algorithm using the ideal theoretic representation given here was presented by Heß in [Heß01]. It relies on the equality

$$L(D) = I^{\text{fin}}(-D) \cap I^{\infty}(-D)_\infty$$

and computes this intersection with a reduction algorithm. In short, one computes a $k[\mathrm{x}]$-basis $v_1, \ldots, v_r$ of $I^{\text{fin}}(-D)$ and integers $d_1, \ldots, d_r$ such that $\mathrm{x}^{d_1} v_1, \ldots, \mathrm{x}^{d_r} v_r$ is a $k[\frac{1}{\mathrm{x}}]_{(\frac{1}{\mathrm{x}})}$-basis of $I^{\infty}(-D)_\infty$. A k-basis of $L(D)$ can then easily be deduced. For the computation one considers the coordinate matrix of some $k[\mathrm{x}]$-basis of $I^{\text{fin}}(-D)$ with respect to a $k[\frac{1}{\mathrm{x}}]_{(\frac{1}{\mathrm{x}})}$-basis of $I^{\infty}(-D)_\infty$ and applies a reduction algorithm on this matrix. The computation can be performed in a time which is polynomially bounded in $d \cdot \text{ht}(D) \cdot \log(q)$.

Also, given a function $f \in k(\mathcal{C})^*$, one can compute the associated principal divisor in a time which is polynomially bounded in $d \cdot \deg(f) \cdot \log(q)$.

Divisors over extension fields. Still, let $k := \mathbb{F}_q$, let $K := \mathbb{F}_{q^n}$ with some natural number n, and let $\mathcal{C}'$ be a curve over K. Let $\mathcal{C}'$ again be given by a plane model in $\mathbb{P}^2_K$, as above. Then there are several obvious possibilities to represent ideals of $(x_*\mathcal{O}_\mathcal{C})(\mathbb{A}^1_K)$.

First, one can proceed as above over K, yielding the usual joint and free representations. Second, one can proceed analogously to the above but starting from the (not geometrically irreducible) scheme $\mathcal{C}'$ over k (corresponding to the function field $K(\mathcal{C}')|k$).

Now, let $\mathcal{C}$ again be a curve over k, represented by a plane model. Let $\mathcal{C}' := \mathcal{C}_K$, and let $\alpha_1, \ldots, \alpha_n$ be a k-basis of K. Then there is a third possibility to represent divisors: Let $w_1, \ldots, w_r$ be as above. Then the elements $\alpha_i w_j$ for $i = 1, \ldots, n$ and

$j = 1, \dots, r$ form a k[x]-basis of $(x_*\mathcal{O}_{\mathcal{C}'})(\mathbb{A}^1_K)$, and one can represent fractional ideals of $(x_*\mathcal{O}_{\mathcal{C}'})(\mathbb{A}^1_K)$ via HNF-bases with respect to this basis.

Of course these considerations also apply to the "infinite ideals" and give several possibilities to represent divisors on $\mathcal{C}'$.

One can show that one can change between the two or three joint representations in a time which is polynomially bounded in $d \cdot \mathrm{ht}(D) \cdot \log(q)$.

Now, again, let $\mathcal{C}' = \mathcal{C}_K$, let $\alpha_1, \dots, \alpha_n$ be a k-basis of K with $\alpha_1 = 1$, and let an ideal I of $(x_*\mathcal{O}_{\mathcal{C}})(\mathbb{A}^1_K)$ be given in joint or free representation as described above. Then one immediately obtains the intersection of I with $k(\mathcal{C})$, again in joint or free representation. In particular, given a divisor D on $\mathcal{C}'$ in free representation, one can compute its norm with respect to the covering $\mathcal{C}' \longrightarrow \mathcal{C}$ in a time which is polynomially bounded in $d \cdot \mathrm{ht}(D) \cdot \log(q)$.

Divisor classes. Let us recall the following definition (see [Heß01]):

Definition 2.4. Let D_0 be a divisor on $\mathcal{C}$ of degree ≥ 1, and let $\tilde{D}$ be an effective divisor on $\mathcal{C}$. Then $\tilde{D}$ is *(maximally) reduced along* D_0 if the linear system $|\tilde{D} - D_0|$ is empty.

Now, let D be any divisor on $\mathcal{C}$, and let $\tilde{D}$ be an effective divisor reduced along D_0 such that $D \sim \tilde{D} + rD_0$ for some $r \in \mathbb{Z}$. Then $\tilde{D}$ is called a *reduction* of D along D_0.

One can easily see that the set of reductions of D along D_0 form a complete linear system. Moreover, if D_0 is a k-rational point of $\mathcal{C}$, then the reduction is unique.

Remark 2.5. It does not suffice that D_0 has degree 1 (without the condition of being effective) in order that the reduction be unique. Here is a counterexample.

Let $\mathcal{C}$ be a hyperelliptic curve with a k-rational point P which is not a Weierstraß point. Let $\tilde{D} := P + \iota(P)$ and let $D_0 := 2\,\iota(P) - P$, where ι is the hyperelliptic involution. Then $\tilde{D} - D_0 = 2P - \iota(P)$, which is not linearly equivalent to a point. Thus $\tilde{D}$ is reduced along D_0, but $|\tilde{D}|$ has dimension 1.

To represent divisor classes, we fix a divisor D_0 of degree ≥ 1 and represent a class a by a corresponding reduced divisor and the degree of a. For computational purposes one can, for example, define D_0 as the canonical divisor $\mathrm{div}(dx)$. With the formula

$$\mathrm{div}(dx) = -2(x)_- + R,$$

where R is the ramification divisor of $x : \mathcal{C} \longrightarrow \mathbb{P}^1_k$, this divisor can be computed in a time which is polynomially bounded in $d \cdot \log(q)$. Generally, if the height of D_0 is polynomially bounded in d, then the arithmetic in $\mathrm{Cl}^0(\mathcal{C})$ can be performed in a time which is polynomially bounded in $d \cdot \log(q)$, provided divisors are represented in joint ideal representation. For the free representation, the computation can be performed in an expected time which is polynomially bounded in $d \cdot \log(q)$.

Note that there always exists a divisor of degree 1 on $\mathcal{C}$. In fact, by the Hasse-Weil bound, for every $m \geq \lceil 2\log_q(2g) \rceil$, there is a $\mathbb{F}_{q^m}$-rational point on $\mathcal{C}$ and thus a closed point on $\mathcal{C}$ whose degree divides m (cf. [Heß05, Proposition 32]). One can compute two closed points with degrees dividing $\lceil 2\log_q(2g) \rceil$ and $\lceil 2\log_q(2g) \rceil + 1$ in an expected time which is polynomially bounded in $d \cdot \log(q)$. These then give rise to a divisor of height $\lceil 2\log_q(2g) \rceil + 1$ and degree 1.

For fixed genus g, one can easily determine if there exists a k-rational point on $\mathcal{C}$ and, if so, compute such a point in an expected time which is polynomially bounded in $\log(q)$. If one defines D_0 as such a point, then—as already mentioned—the representation of degree 0 divisor classes by reduced divisors is unique. In the algorithm for Theorem 1 we employ such a representation.

Let us finally consider the important case that d is polynomially bounded in g. Then in the statements above, $d \cdot \log(q)$ can of course be substituted by $g \cdot \log(q)$. Moreover, as shown in [LMD90], one has

$$\# \operatorname{Cl}^0(\mathcal{C}) \geq \frac{(q-1)^2}{(q+1)\cdot q} \cdot \frac{1}{g+1} \cdot q^g \geq \frac{1}{6} \cdot \frac{1}{g+1} \cdot q^g, \tag{2}$$

and this implies that $d \cdot \log(q)$ can then also be substituted by $\log(\# \operatorname{Cl}^0(\mathcal{C}))$. In particular, if the degree of the divisor D_0 is also polynomially bounded in g, the arithmetic in the degree 0 class group can then be performed in a time which is polynomially bounded in $\log(\# \operatorname{Cl}^0(\mathcal{C}))$.

Representation of input data. We can now describe how the input data in the algorithms for the theorems are represented.

The curve is always given by a plane model of degree $\mathcal{O}(g)$; in Theorem 1 it is therefore given by a plane model of bounded degree. We fix a divisor D_0 of degree 1 and height $\mathcal{O}(g)$, and we represent divisor classes by along D_0 reduced divisors. The divisors in turn are given in free or joint ideal representation as described above.

Unique representation and generic algorithms. For some purposes, for example, for the baby-step-giant-step algorithm, one requires a unique representation of elements of $\operatorname{Cl}^0(\mathcal{C})$ by bit strings.

One possibility to achieve this is to consider a constant field extension such that the curve has a rational point. For example, as mentioned above, with $m := \lceil 2\log_q(2g) \rceil$, the curve $\mathcal{C}_{\mathbb{F}_{q^m}}$ has a $\mathbb{F}_{q^m}$-rational point, and with a randomized algorithm, one can compute such a point P_0 in an expected time which is polynomially bounded in $d \cdot \log(q)$. All elements of $\operatorname{Cl}^0(\mathcal{C}_{\mathbb{F}_{q^m}})$, thus, in particular, those of the subgroup $\operatorname{Cl}^0(\mathcal{C})$, can then uniquely be represented by along P_0 reduced divisors.

Given an instance of the discrete logarithm problem as described above, we would like to compute the discrete logarithm deterministically in a time of $\tilde{\mathcal{O}}(\# \operatorname{Cl}^0(\mathcal{C})^{\frac{1}{2}})$.

In the case of genus 1 curves, D_0 is linearly equivalent to a unique $\mathbb{F}_q$-rational point P_0. One can therefore apply the baby-step-giant-step algorithm directly, and one obtains the desired result.

We therefore consider the case that the genus is ≥ 2. Again, let p be the characteristic. In this case we first compute a field extension $\lambda|\mathbb{F}_q$ of degree $\leq \operatorname{ht}(D_0)$ and a λ-rational point P_0 on $\mathcal{C}_\lambda$. Here the field extension is represented by a multiplication table. We then represent divisor classes by along P_0 reduced divisors and again apply the baby-step-giant-step algorithm. The baby-step-giant-step algorithm has a running time of $\tilde{\mathcal{O}}(\# \operatorname{Cl}^0(\mathcal{C})^{\frac{1}{2}})$, and we have to show that the computation of λ and P_0 can be performed (deterministically) in this time as well.

For this, we proceed as follows: First we compute the free representation of D_0, and we fix a prime divisor E occurring in D_0. As mentioned above, this can be achieved deterministically in a time of $p \cdot (g \cdot \log(q))^{\mathcal{O}(1)}$. Note also that $\deg(E) \leq \operatorname{ht}(D_0) \in \mathcal{O}(g)$. Assume now that E corresponds to a "finite" ideal, that is, to an ideal in $(x_*\mathcal{O}_\mathcal{C})(\mathbb{A}^1_{\mathbb{F}_q})$ (the other case is similar); let $\mathfrak{p}$ be this ideal. An

HNF-basis for $\mathfrak{p}$ easily gives an $\mathbb{F}_q$ basis of $(x_*\mathcal{O}_\mathcal{C})(\mathbb{A}^1_\lambda)/\mathfrak{p}$ and a multiplication table; let λ be this field. Then E as a divisor on $\mathcal{C}_\lambda$ splits completely. So we compute the free representation of E as a divisor of $\mathcal{C}_\lambda$ and fix one of the points occurring in it. This can again be achieved (deterministically) in a time of $p \cdot (g \cdot \log(q))^{\mathcal{O}(1)}$.

We remark that for genus 1 curves over prime fields the assumption that D_0 has degree 1 is crucial. If we drop this assumption, we do not know how to perform the computation (deterministically) in the desired time.

3. Index calculus for curves of fixed genus

3.1. **Overview.** In this section we present an index calculus algorithm with double large prime variation for curves of a fixed genus $g \geq 2$, leading to Theorem 1.

The input of the algorithm consists of a tuple $(\mathcal{C}/\mathbb{F}_q, a, b)$, where $\mathcal{C}$ is a curve of genus g over $\mathbb{F}_q$, $a, b \in \mathrm{Cl}^0(\mathcal{C})$ and $b \in \langle a \rangle$. The representation of the input follows the description given in the previous section: The curve is represented by a plane model, a divisor D_0 of degree 1 is fixed and a, b are represented by along D_0 reduced divisors which in turn are given in free or joint ideal representation as described above. Here the degree of the plane model as well as the height of D_0 are bounded by some constant.

We now give an outline of the proof of Theorem 1.

First we give an index calculus algorithm with double large prime variation which gives rise to the following proposition.

Proposition 3.1. *Let us fix some $g \geq 2$. Then there exists a randomized algorithm such that the following holds: Upon input of a curve $\mathcal{C}/\mathbb{F}_q$ of genus g, elements $a, b \in \mathrm{Cl}^0(\mathcal{C})$ with $b \in \langle a \rangle$ and a system $c_1, \ldots, c_u$ of elements of $\mathrm{Cl}^0(\mathcal{C})$, if the algorithm terminates it outputs the discrete logarithm of b with respect to a. Moreover, if $c_1, \ldots, c_u$ forms a generating system and u is polynomially bounded in $\log(q)$, the expected running time of the algorithm is in $\tilde{\mathcal{O}}(q^{2-\frac{2}{g}})$.*

The algorithm has storage requirements of $\tilde{\mathcal{O}}(q^{1-\frac{1}{g}+\frac{1}{g^2}})$. More precisely, the algorithm uses only the first $\tilde{\mathcal{O}}(q^{1-\frac{1}{g}+\frac{1}{g^2}})$ registers, and each register always contains elements whose bit-length is polynomially bounded in $\log(q)$.

The algorithm is given in the next subsection and the analysis in subsection 3.3.

Then we show the that there is an efficient algorithm which outputs a small system of divisor classes of degree 0 which generates the degree 0 class group with high probability (see subsection 3.5):

Proposition 3.2. *Let us fix some $g \geq 1$. Then there exists a randomized algorithm such that the following holds: Upon input of a curve $\mathcal{C}/\mathbb{F}_q$ of genus g and a divisor D_0 of degree 1 as above, the algorithm computes a system of random elements $c_1, \ldots, c_u$ of $\mathrm{Cl}^0(\mathcal{C})$, represented by along D_0 reduced divisors, where $u := e\ell$ with $e := \lceil \log_2(\# \mathrm{Cl}^0(\mathcal{C})) \rceil$ and $\ell := \lceil \log_2(e) + 1 \rceil$. Moreover, the expected running time is polynomially bounded in d and $\log(q)$, and with a probability $\geq \frac{1}{2}$, the system $c_1, \ldots, c_u$ is a generating system of $\mathrm{Cl}^0(\mathcal{C})$.*

A problem is, however, that we do not know how one can in a sufficiently efficient way certify that the output is indeed a generating system. As a work-around, we proceed as follows: We assume that we have a generating system and apply the index calculus algorithm. We stop the computation if it has not terminated within a predefined time.

An obvious consequence of Proposition 3.1 follows:

Proposition 3.3. *Let us fix some $g \geq 2$. Then there exists a randomized algorithm such that the following holds: Upon input of a curve $\mathcal{C}/\mathbb{F}_q$ of genus g, elements $a, b \in \mathrm{Cl}^0(\mathcal{C})$ with $b \in \langle a \rangle$ and a system $c_1, \dots, c_u$ of elements in $\mathrm{Cl}^0(\mathcal{C})$ with u polynomially bounded in $\log(q)$, the algorithm either outputs "failure" or the discrete logarithm of b with respect to a. The running time of the algorithm is in $\tilde{\mathcal{O}}(q^{2-\frac{2}{g}})$, and if $c_1, \dots, c_u$ is a generating system, the probability of failure is $\leq \frac{1}{2}$. The algorithm has storage requirements of $\tilde{\mathcal{O}}(q^{1-\frac{1}{g}+\frac{1}{g^2}})$.*

Indeed, let us fix a RAM Π satisfying the claim in Proposition 3.1, and let $\delta \in \mathbb{N}$ such that the RAM terminates in an expected time of $q^{2-\frac{2}{g}} \cdot \log(q)^\delta$ if applied to an instance as in Proposition 3.1. We apply this RAM with the input of Proposition 3.3 and terminate the execution if $\lceil 2 \cdot q^{2-\frac{2}{g}} \cdot \log(q)^\delta \rceil$ commands are executed (or in other words, if the *uniform* running time is $\lceil 2 \cdot q^{2-\frac{2}{g}} \cdot \log(q)^\delta \rceil$), provided Π has not terminated at this point in time. (For this, we insert a "control unit".) Note that then the running time of Π (as always measured with the logarithmic cost measure) is $\geq \lceil 2 \cdot q^{2-\frac{2}{g}} \cdot \log(q)^\delta \rceil$. By Markov's bound we have: If $c_1, \dots, c_u$ is a generating system, with a probability of $\geq \frac{1}{2}$, the algorithm outputs the discrete logarithm of b with respect to a.

By assumption, each operation performed by Π has a running time which is polynomially bounded in $\log(q)$. Therefore, the total running time is in $\tilde{\mathcal{O}}(q^{2-\frac{2}{g}})$. □

The *proof of Theorem* 1 is now easy:

Let $g \geq 2$ be fixed. Now given an instance $(\mathcal{C}, a, b)$ with $g(\mathcal{C}) = g$ as described above, we proceed as follows: First we apply an algorithm for which Proposition 3.2 holds; let the result be $c_1, \dots, c_u$. Then we apply an algorithm for which Proposition 3.3 holds to $\mathcal{C}, a, b$ and the system $c_1, \dots, c_u$. The expected running time is then in $\tilde{\mathcal{O}}(q^{2-\frac{2}{g}})$, and moreover, the probability of failure is $\leq \frac{3}{4}$. This implies Theorem 1. □

3.2. **The index calculus algorithm.** Again, let $g \geq 2$ be fixed. We now describe an index calculus algorithm which leads to Proposition 3.1. As already stated, the algorithm uses double large prime variation. We do so by computing a tree of large prime relations. The main challenge resides in controlling the growth of the tree of large prime relations as well as its depth, that is, the maximal distance of any vertex to the root. This task is more difficult than for hyperelliptic curves in imaginary quadratic representation, where one has a concrete description of the effective divisors which are reduced along the point at infinity, and one knows that the growth process is very regular.

We now give an overview of the algorithm. Subroutines for the various steps are discussed later in greater detail.

Broadly speaking, we proceed as follows: First we compute the group order and its factorization. We fix an $\mathbb{F}_q$-rational point P_0 and a factor base $\mathcal{F}$. We construct a tree of large prime relations, using a "stagewise" approach. After a tree of a suitable size has been constructed, we use the tree to generate relations over the factor base. The main idea here is that we can—via the tree—substitute large primes occurring in relations by sums of factor base elements. Finally, we solve the discrete logarithm problem via sparse linear algebra.

More precisely, the factor base is a set $\mathcal{F} \subseteq \mathcal{C}(\mathbb{F}_q) - \{P_0\}$ of size $\lceil q^{1-\frac{1}{g}} \rceil$, and the set of so-called "large primes" is $\mathcal{L} := \mathcal{C}(\mathbb{F}_q) - (\mathcal{F} \cup \{P_0\})$. Now the tree of large prime relations is a tree T whose set of vertices is contained in $\mathcal{L} \,\dot{\cup}\, \{*\}$. In the following, we denote the set of vertices of T also by T. How the tree is computed is discussed below.

Apart from the computation of the tree of large prime relations, the algorithm is closely related to the algorithm in [EG02]. A relatively minor difference is that in [EG02] it is assumed that the group is cyclic or a basis of it is known whereas we use only a generating system.

The input data for the algorithm are represented as described at the end of the previous section.

The algorithm for curves of a fixed genus g.

```
Input: A curve $\mathcal{C}/\mathbb{F}_q$ of genus $g$, elements $a, b \in \mathrm{Cl}^0(\mathcal{C})$ with $b \in \langle a \rangle, c_1, \ldots, c_u \in \mathrm{Cl}^0(\mathcal{C})$, given by along $D_0$ reduced divisors.

Output: The discrete logarithm of $b$ with respect to $a$.

1. Compute $N \longleftarrow \#\,\mathrm{Cl}^0(\mathcal{C})$ and factorize $N$. (Let $N = \prod_{i=1}^{v} \ell_i^{e_i}$ with $e_i \in \mathbb{N}$ and pairwise distinct prime elements $\ell_i$.)
2. Enumerate $\mathcal{C}(\mathbb{F}_q)$. Choose an $\mathbb{F}_q$-rational point $P_0$ and a factor base $\mathcal{F}=\{F_1, F_2, \ldots, F_k\} \subseteq \mathcal{C}(\mathbb{F}_q) - \{P_0\}$ of size $\lceil q^{1-\frac{1}{g}} \rceil$. (If this is not possible, terminate.) Represent $a, b, c_1, \ldots, c_u$ by along $P_0$ reduced divisors.
3. Construct a labeled rooted tree $T$ with root $*$ whose set of vertices is contained in $\mathcal{L} \,\dot{\cup}\, \{*\}$, as described below.
4. 4.1. Let $t \longleftarrow \lceil \log_2(k+u) + \log_2(2\log_2(N)) \rceil + 1$
   4.2. Construct matrices $R \in (\mathbb{Z}/N\mathbb{Z})^{(2(k+u)\cdot t)\times k}$ and $S \in (\mathbb{Z}/N\mathbb{Z})^{(2(k+u)\cdot t)\times u}$ in sparse representation as well as vectors $\underline{\alpha}, \underline{\beta} \in (\mathbb{Z}/N\mathbb{Z})^{2(k+u)\cdot t}$ as follows:
        4.2.1. For $i = 1, \ldots, (k+2u)\cdot t$ do
                   Choose uniformly and independently randomly $\alpha, \beta, s_1, \ldots, s_u \in \mathbb{Z}/N\mathbb{Z}$ and compute the unique along $P_0$ reduced divisor $D$ with $[D] - \deg(D)\cdot[P_0] = \sum_j s_j c_j + \alpha a + \beta b$ in free representation.
               Until $D$ splits into elements of $\mathcal{F} \cup T$.
               Use the tree $T$ to substitute these elements by sums of multiples of elements of $\mathcal{F}$. Let
                   $$\sum_j r_{i,j}[F_j] - r_i[P_0] = \sum_j s_{i,j} c_j + \alpha_i a + \beta_i b$$
               with $r_i = \sum_j r_{i,j}$ be the relation generated.
        4.2.2. For $i_1 = 1, \ldots, k$
                   For $i_2 = 1, \ldots, t$
                       Let $i \longleftarrow (i_1 - 1)\cdot t + i_2 + (k+2u)\cdot t$.
                       Choose uniformly and independently randomly $\alpha, \beta, s_1, \ldots, s_u \in \mathbb{Z}/N\mathbb{Z}$ and compute the unique along $P_0$ reduced divisor $D$ with $[D] - \deg(D)\cdot[P_0] = [F_{i_1}] - [P_0] + \sum_j s_j c_j + \alpha a + \beta b$.
                   Until $D$ splits into elements of $\mathcal{F} \cup T$.
                   Use the tree $T$ to substitute these elements by sums of multiples of elements of $\mathcal{F}$. Let
                       $$\sum_j r_{i,j}[F_j] - r_i[P_0] = \sum_j s_j c_j + \alpha_i a + \beta_i b$$
```

with $r_i = \sum_j r_{i,j}$ be the relation generated.

5. Try to compute a row vector $\underline{\gamma} \in (\mathbb{Z}/N\mathbb{Z})^{1\times(2(k+u)\cdot t)}$ with $\underline{\gamma}(R|S) = 0$ and $[\underline{\gamma}]_\ell \neq 0$ for all prime divisors ℓ of N with an algorithm for sparse linear algebra as discussed below.[2] If this fails, go back to Step 4.
6. If $\sum_i \gamma_i\beta_i \in (\mathbb{Z}/N\mathbb{Z})^*$, let $\xi := -\dfrac{\sum_i \gamma_i\alpha_i}{\sum_i \gamma_i\beta_i}$, otherwise go back to Step 4.
7. Compute $\mathrm{ord}(a)$, using the factorization of N. Output the unique non-negative number $x \in \{0, \ldots, \mathrm{ord}(a)-1\}$ with $[x]_{\mathrm{ord}(a)} = [\xi]_{\mathrm{ord}(a)}$.

It is immediate that the algorithm is correct, that is, if it terminates, it outputs the discrete logarithm of b with respect to a.

As already mentioned, details on the various steps of the algorithm will be given below. However, merely under an assumption on the linear algebra algorithm, we can already now give an upper bound on the number of expected iterations of Steps 4–6, conditionally to any outcome of Step 3:

Proposition 3.4. *Let the linear algebra algorithm be such that the following holds: If applied to a matrix M in sparse representation over $\mathbb{Z}/N\mathbb{Z}$:*

- *the algorithm terminates in a finite expected time, and it either outputs a vector $\underline{\gamma}$ over $\mathbb{Z}/N\mathbb{Z}$ with $\underline{\gamma}M = 0$ and $[\underline{\gamma}]_\ell \neq \underline{0}$ for all prime divisors ℓ of N or it outputs "failure";*
- *if, moreover, the matrix M has full column rank, the algorithm succeeds (that is, outputs a vector as desired) with a probability of $\Omega(1)$.*

If then $c_1, \ldots, c_u$ forms a generating system, conditionally to any outcome of Step 3, the probability that the algorithm fails in Step 5 or Step 6 (that is, it returns to Step 4) is in $\mathcal{O}(\log\log(q))$.

This proposition is essentially proven in [EG02]. We recall two major ingredients of the proof.

First, one can show that the probability that the matrix $(R|S)$ has full column rank is in $\Omega(1)$. This follows from Lemma 4.1 in [Pom87] and the subsequent remarks. We note that the proof in [Pom87] is very sketchy; a more complete argument is given in [Heß05] (see part (i) of [Heß05, Lemma 64]).

Moreover, the random vector $\underline{\beta}$ is independent of $(R|S)$. This implies that the probability that $\sum_i \gamma_i\beta_i$ is invertible is $\frac{\varphi(N)}{N}$, which is in $\Omega(\frac{1}{\log\log(N)}) = \Omega(\frac{1}{\log\log(q)})$ (see [RS62]). □

Subroutines. We now give some more information on various steps of the algorithm.

Step 1 – Computation of the group order and factorization. The L-polynomial of a curve over $\mathbb{F}_q$ given by a plane model of bounded degree can be computed (with a deterministic algorithm) in a time which is polynomially bounded in $\log(q)$. (This follows from [Pil91, Theorem H] which in turn relies on Pila's extension of the point counting algorithm by Schoof ([Sch85]) to abelian varieties ([Pil90]).) This means, in particular, that the order of the degree 0 class group can be computed in polynomial time in $\log(q)$.

[2] For an integer a and a natural number n we denote the residue class of a in $\mathbb{Z}/n\mathbb{Z}$ by $[a]_n$.

It is well known that an integer N can be factored in subexponential time in $\log(N)$; see for example [Pom87].

Step 2 – Construction of the factor base. As assumed in the previous section, let $x : \mathcal{C} \longrightarrow \mathbb{P}^1_k$ be separable. Then one can iterate over all points of $\mathcal{C}(\mathbb{F}_q)$ in an expected time of $\tilde{\mathcal{O}}(q)$ as follows: One iterates over all points $P \in \mathbb{P}^1(\mathbb{F}_q)$ and factorizes the ideal $x^{-1}(P)$ (that is, one computes a free representation). For each point P, this step requires an expected time which is polynomially bounded in $\log(q)$. In total Step 2 can be performed in an expected time of $\tilde{\mathcal{O}}(q)$.

Step 3 – Generation of the tree of large prime relations. The tree of large prime relations is a labeled rooted tree whose vertex set is contained in $\mathcal{L} \dot{\cup} \{*\}$ with root $*$. It is constructed as follows:

We repeatedly choose uniformly randomly $s_1, \ldots, s_u \in \mathbb{Z}/N\mathbb{Z}$ and compute the along P_0 reduced effective divisor D in free representation with

$$[D] - \deg(D) \cdot [P_0] = \sum_{j=1}^{u} s_j\, c_j, \tag{3}$$

where $P_0 \in \mathcal{C}(\mathbb{F}_q)$ is the point computed above.[3] Note that as mentioned in the previous section, the computation of D is possible in an expected time which is polynomially bounded in $\log(q)$.

Following [GTTD07] and other works, we define:

Definition 3.5. A relation involving only input elements and factor base elements, that is, a relation of the form $\sum_j r_j[F_j] = \sum_j s_j c_j + \alpha a + \beta b$ is called a *Full relation.* A relation which additionally involves one large prime, that is, a relation of the form $\sum_j r_j[F_j] + r_P[P] = \sum_j s_j c_j + \alpha a + \beta b$ with $P \in \mathcal{L}$ and $r_P \neq 0$ is called an *FP relation.* A relation which involves two large primes, that is, a relation of the form $\sum_j r_j[F_j] + r_P[P] + r_Q[Q] = \sum_j s_j c_j + \alpha a + \beta b$ with $P, Q \in \mathcal{L}$, $P \neq Q$ and $r_P, r_Q \neq 0$ is called a *PP relation.* Here $r_j, r_P, r_Q \in \mathbb{Z}/N\mathbb{Z}$.

As usual, during the construction of the tree, FP relations are stored as labeled edges connecting the large prime with the root $*$, and PP relations are stored as labeled edges connecting the two large primes.

As already mentioned, we construct the tree in *stages.* During each stage we only attach edges to the tree which are connected to vertices constructed *in the previous stages.* In Stage 1, we attach $\lceil q^{1-1/g} \rceil$ edges coming from FP relations to the root $*$. Thereafter, we terminate Stage s and start Stage $s+1$ whenever the tree has $2^{s-1} \cdot \lceil q^{1-1/g} \rceil$ edges.

The construction of the tree is abandoned if a predefined number of edges $N_{\max}$ is reached. We could for example set $N_{\max} := \lceil q/4 \rceil$. We will, however, argue in the analysis of the algorithm in the next subsection that $N_{\max} := \lceil q^{1-1/g+1/g^2} \rceil$ suffices. This smaller value of $N_{\max}$ only lowers the time for the construction of the tree by a constant factor but decreases the storage requirements substantially. This is analogous to the situation in [GTTD07].

Altogether, we have the following procedure for construction of a suitable tree of large prime relations. In the procedure we construct a labeled tree called T.

[3] Note here that if we represent divisor classes by along P_0 reduced divisors in free representation, to compute the divisor D is the same as computing the sum $\sum_{j=1}^{u} s_j\, c_j$ in the degree 0 class group.

The edges of the tree are labeled with the corresponding relations; the vertices are labeled too, namely with the stage at which they inserted into the tree. We denote the subtree of T which has been constructed until (including) stage s by T_s. In other words, a vertex of T occurs in T_s if and only if its label is $\leq s$.

Procedure: Construction of the tree of large prime relations.

```
Construct a labeled rooted tree $T$ with vertex set contained in $\mathcal{L} \dot{\cup} \{*\}$ as follows:
Let $T$ consist only of the root $*$, labeled with 0.
Let $N_{\max} \longleftarrow \lceil q^{1-1/g+1/g^2} \rceil$.
Let $s \longleftarrow 1$.
Repeat
    Repeat
        Choose $s_1, \ldots, s_u \in \mathbb{Z}/N\mathbb{Z}$ uniformly and independently at random.
        Compute the along $P_0$ reduced divisor $D$ in free representation with $[D] - \deg(D) \cdot [P_0] = \sum_j s_j c_j$.
        If $D$ splits as $D = \sum_j r_j F_j + Q$ with $Q \in \mathcal{L} - (\mathcal{F} \cup T)$,
            insert an edge from $*$ to $Q$ into $T$, labeled with $(r_j)_j$ (in sparse representation).
        If $D$ splits as $D = \sum_j r_j F_j + r_P P + Q$ with $P \in T_{s-1}, Q \in \mathcal{L} - (\mathcal{F} \cup T)$ and $r_P > 0$,
            insert an edge from $P$ to $Q$ into $T$, labeled with $(r_j)_j$ and $r_P$.
        In both cases label $Q$ with $s$ and the edge with $(r_j)_j$ (in sparse representation).
    Until $T$ contains $\min\{2^{s-1} \cdot \lceil q^{1-1/g} \rceil, N_{\max}\}$ edges.
    If the number of edges equals $N_{\max}$, STOP.
    Let $s \longleftarrow s + 1$.
```

This construction of the tree guarantees that the depth of the tree is always in $\mathcal{O}(\log(q))$ (see also inequality (7) in the next subsection). The main difficulty of the analysis of the procedure resides in proving that a tree of sufficient size can be constructed in an expected time of $\tilde{\mathcal{O}}(q^{2-2/g})$. This is verified in the next subsection.

Step 5 – Linear algebra. For the linear algebra computation we use an algorithm which satisfies the following condition (and thus also Proposition 3.4).

If applied to a matrix $M \in (\mathbb{Z}/N\mathbb{Z})^{m \times n}$ with $m \geq n$ and weight ω in sparse representation, then

- the algorithm always terminates in an expected time of $\tilde{O}(n \cdot (m + \omega) \cdot \log(mN)^{\mathcal{O}(1)})$, and it either outputs a vector $\underline{\gamma}$ over $\mathbb{Z}/N\mathbb{Z}$ with $\underline{\gamma} M = 0$ and $[\underline{\gamma}]_\ell \neq \underline{0}$ for all prime divisors ℓ of N or it outputs "failure",
- if, moreover, the matrix M has full column rank, the algorithm succeeds with a probability of $\Omega(1)$.

Here the *weight* of a matrix is the number of non-zero entries.

Such an algorithm is given in [EG02, Section 4].

Step 7 – Computation of the order of *a*. The computation of $\operatorname{ord}(a)$ can be performed efficiently (in polynomial time in $\log(N) \in \mathcal{O}(\log(q))$) along the following lines:

As in the algorithm, let $N = \prod_{i=1}^{v} \ell_i^{e_i}$ with $e_i \in \mathbb{N}$ and pairwise distinct prime numbers ℓ_i. Now let $L_i := \frac{N}{\ell_i^{e_i}}$, and let $o_i := \min\{j \in 0, \ldots, e_i \mid \ell_i^j L_i \cdot a = 0\}$ for $i = 1, \ldots, v$. Then $\prod_{i=1}^{v} \ell_i^{o_i}$ is the order of a.

3.3. **Analysis of the index calculus algorithm.** We now show that with the algorithm outlined in the previous subsection, one can compute a solution to the discrete logarithm problem for curves of the fixed genus g in an expected time of $\tilde{\mathcal{O}}(q^{2-2/g})$, provided that the system $c_1, \ldots, c_u$ is a generating system and u is polynomially bounded in $\log(q)$.

Steps 1 and 2 – Computation of the group order, factorization, construction of the factor base. We have already argued that one can perform these steps in an expected time of $\tilde{O}(q)$.

Step 3 – Construction of the tree of large prime relations. The analysis of the growth of the tree relies crucially on the following proposition.

Proposition 3.6. *For curves of fixed genus g over finite fields $\mathbb{F}_q$, the number of special effective divisors of degree g is in $\mathcal{O}(q^{g-1})$.*

Recall that an effective divisor D is called *special* if the linear system $|K - D|$ is non-empty, where K is a canonical divisor. Note that by the Riemann-Roch theorem, an effective divisor of degree g is non-special if and only if the linear system $|D|$ merely contains D itself.

Note that map $D \mapsto [D]$ gives an injection from the set of non-special divisors of degree g into the set of divisor classes of degree g, and therefore, the map $D \mapsto [D - g \cdot P_0]$ gives an injection from the set of non-special divisors of degree g into the degree 0 divisor class group. Then we can apply the bijection between the degree 0 class group and the set of along P_0 reduced divisors.

Explicitly, let D be a non-special effective divisor of degree g, and let D' be the unique effective divisor of minimal degree with $D' + (\deg(D) - \deg(D')) \cdot P_0 = D$. Then D' is reduced along P_0, and it is the along P_0 reduced divisor which represents the class $[D - g \cdot P_0]$.

We assume that Proposition 3.6 is well known to many experts in curves and function fields. For the lack of a suitable reference we give a proof in the next subsection. Note that a straightforward application of the Hasse-Weil bound merely gives that the number in question is in $O(q^{g-1/2})$.

This proposition makes it possible to discard all special divisors in the analysis of the construction of the tree of large prime relations.

Let $C > 0$ be such that for all curves of genus g over all finite fields $\mathbb{F}_q$ the number of special divisors of degree g is $\leq C \cdot q^{g-1}$.

As in the previous subsection, let $N_{\max} := \lceil q^{1-1/g+1/g^2} \rceil$ be the number of edges (that is, the number of vertices different from $*$) at which the construction of the tree is stopped.

The conditions

$$N_{\max} + \#\mathcal{F} \leq q/4, \qquad \#(\mathcal{C}(\mathbb{F}_q) - \{P_0\}) \in [\max\{q^{1-\frac{1}{g}}, q/2\}, 2q],$$
$$\#\operatorname{Cl}^0(\mathcal{C}) \leq 2q^g, \qquad q \geq (4 \cdot g! \cdot C)^g,$$

hold for q large enough; we assume that they are satisfied.

Note that by our assumption that $c_1, \dots, c_u$ generate $\mathrm{Cl}^0(\mathcal{C})$, if $s_1, \dots, s_u$ are uniformly distributed random elements from $\mathbb{Z}/N\mathbb{Z}$, $\sum_i s_i c_i$ is uniformly distributed in $\mathrm{Cl}^0(\mathcal{C})$. This means that the divisor D in (3) is uniformly distributed in the set of all effective divisors which are reduced along P_0.

By our assumptions on q, we always have

$$\#(\mathcal{C}(\mathbb{F}_q) - (T \cup \mathcal{F} \cup \{P_0\}) \geq q/2 - q/4 = q/4. \tag{4}$$

Let $\mathrm{Div}^g(\mathcal{C})$ be the set of effective divisors of degree g on $\mathcal{C}$, and let $\mathrm{Div}^{g,ns}(\mathcal{C})$ (resp. $\mathrm{Div}^{g,\mathrm{s}}(\mathcal{C})$) be the subset of non-special (resp. special) effective divisors of degree g.

Let us first assume that we are still in Stage 1, that is, only relations with one large prime (not yet in the tree) are considered.

Let us thus assume we are given the tree T with $< \lceil q^{1-1/g} \rceil$ edges. We want to bound the expected number of relations (3) needed until a new edge is inserted into the tree.

Let

$$\begin{aligned} \mathcal{D} := \big\{ P_1 + \cdots + P_g \in \mathrm{Div}^g(\mathcal{C}) \mid \forall i = 1, \dots, g-1 : P_i \in \mathcal{F}, \\ P_g \in \mathcal{C}(\mathbb{F}_q) - (T \cup \mathcal{F} \cup \{P_0\}) \big\}, \\ \mathcal{D}^{ns} := \mathcal{D} \cap \mathrm{Div}^{g,ns}(\mathcal{C}). \end{aligned}$$

Note that any divisor $D \in \mathcal{D}^{ns}$ is reduced along P_0 (because P_0 is not contained in the support of D and the linear system $|D|$ consists merely of D). If a divisor $D = P_1 + \cdots + P_g$ as in the set $\mathcal{D}^{ns}$ appears in a relation (3), a new edge is inserted into the tree. (Other divisors might also lead to new edges: We ignore special divisors and we ignore divisors of degree $< g$.)

We have

$$\begin{aligned} \#\mathcal{D} &= \tbinom{\#\mathcal{F}+g-2}{g-1} \cdot \#\big(\mathcal{C}(\mathbb{F}_q) - (T \cup \mathcal{F} \cup \{P_0\})\big) \geq \tfrac{\#\mathcal{F}^{g-1}}{(g-1)!} \cdot q/4 \qquad \text{by (4)} \\ &\geq \tfrac{1}{4(g-1)!} \cdot q^{\frac{(g-1)^2}{g}} \cdot q = \tfrac{1}{4(g-1)!} \cdot q^{\frac{g^2-g+1}{g}} = \tfrac{1}{4(g-1)!} \cdot q^{g-1+\frac{1}{g}}. \end{aligned} \tag{5}$$

By our assumption that $q \geq (4 \cdot g! \cdot C)^g$, we have

$$\#\,\mathrm{Div}^{g,\mathrm{s}}(\mathcal{C}) \leq Cq^{g-1} \leq \frac{1}{4g!} \cdot q^{g-1+\frac{1}{g}} \leq \frac{1}{8(g-1)!} \cdot q^{g-1+\frac{1}{g}}. \tag{6}$$

Inequalities (5) and (6) imply

$$\#\mathcal{D}^{ns} \geq \frac{1}{8(g-1)!} \cdot q^{g-1+\frac{1}{g}}.$$

Together with our assumption that $\#\,\mathrm{Cl}^0(\mathcal{C}) \leq 2q^g$, this implies that the probability that a relation (3) enlarges the tree is

$$\geq \frac{\#\mathcal{D}^{ns}}{\#\,\mathrm{Cl}^0(\mathcal{C})} \geq \frac{1}{16(g-1)!} \cdot q^{-(1-\frac{1}{g})}.$$

The expected number of relations (3) which have to be considered until the tree is enlarged is thus

$$\leq 16(g-1)! \cdot q^{1-\frac{1}{g}}.$$

This implies that the expected number of tries until the tree has $\lceil q^{1-\frac{1}{g}} \rceil$ edges is

$$\leq 16(g-1)! \cdot q^{1-\frac{1}{g}} \cdot \lceil q^{1-\frac{1}{g}} \rceil \leq 16(g-1)! \cdot (q+1)^{2-\frac{2}{g}}.$$

We now assume that $s \geq 2$ and a tree T with $< 2^{s-1} \cdot \lceil q^{1-1/g} \rceil$ edges containing a subtree T_{s-1} with $2^{s-2} \cdot \lceil q^{1-1/g} \rceil$ edges, has already been constructed. The task is again to derive a bound on the expected number of relations (3) needed until the tree is enlarged.

Similarly to the above, let

$$\begin{aligned} \mathcal{D} &:= \big\{ P_1 + \cdots + P_g \in \mathrm{Div}^g(\mathcal{C}) \,|\, \forall i = 1, \ldots, g-2 : P_i \in \mathcal{F}, \\ &\qquad\qquad P_{g-1} \in \mathcal{F} \cup T_{s-1},\ P_g \in \mathcal{C}(\mathbb{F}_q) - (T \cup \mathcal{F} \cup \{P_0\}) \big\}, \\ \mathcal{D}^{ns} &:= \mathcal{D} \cap \mathrm{Div}^{g,ns}(\mathcal{C}). \end{aligned}$$

We now have

$$\begin{aligned} \#\mathcal{D} &= \big(\tbinom{\#\mathcal{F}+g-2}{g-1} + \tbinom{\#\mathcal{F}+g-3}{g-2} \cdot \#(T_{s-1} - \{*\}) \big) \cdot \#\big(\mathcal{C}(\mathbb{F}_q) - (T \cup \mathcal{F} \cup \{P_0\})\big) \\ &\geq \big(\tfrac{\#\mathcal{F}^{g-1}}{(g-1)!} + \tfrac{\#\mathcal{F}^{g-2}}{(g-2)!} \cdot 2^{s-2} \cdot q^{1-1/g} \big) \cdot q/4 \\ &\geq \big(\tfrac{1}{(g-1)!} \cdot q^{\frac{(g-1)^2}{g}} + \tfrac{1}{(g-2)!} \cdot 2^{s-2} \cdot q^{\frac{(g-1)^2}{g}} \big) \cdot q/4 \\ &= \big(\tfrac{1}{4(g-1)!} + \tfrac{1}{4(g-2)!} \cdot 2^{s-2} \big) \cdot q^{g-1+\frac{1}{g}}. \end{aligned}$$

Together with (6), this implies

$$\#\mathcal{D}^{ns} \geq \frac{1}{4(g-2)!} \cdot 2^{s-2} \cdot q^{g-1+\frac{1}{g}}.$$

This implies that the probability that a relation as in (3) enlarges the tree is

$$\geq \frac{1}{8(g-2)!} \cdot 2^{s-2} \cdot q^{-(1-\frac{1}{g})}.$$

The expected number of relations as in (3) which have to be considered until the tree is enlarged is thus

$$\leq 8(g-2)! \cdot \frac{1}{2^{s-2}} \cdot q^{1-\frac{1}{g}}.$$

This implies that given any tree T_{s-1} with $2^{s-2} \cdot \lceil q^{1-\frac{1}{g}} \rceil$ edges, the expected number of tries until a tree T with $\min\{2^{s-1} \cdot \lceil q^{1-\frac{1}{g}} \rceil, N_{\max}\}$ edges is constructed is

$$\leq 16(g-2)! \cdot (q+1)^{2-\frac{2}{g}}.$$

We have $s \in \mathcal{O}(\log(q))$ as can be easily seen: During the execution of the procedure we always have for $s \geq 2$

$$2q \geq \#(T - \{*\}) \geq \#(T_{s-1} - \{*\}) = 2^{s-2} \cdot \lceil q^{1-\frac{1}{g}} \rceil,$$

thus

$$s \leq \log_2(q^{\frac{1}{g}}) + 3 = \frac{1}{\log(2) \cdot g} \cdot \log(q) + 3 \in \mathcal{O}(\log(q)). \tag{7}$$

It follows that in total an expected number of $\mathcal{O}(\log(q) \cdot q^{2-\frac{2}{g}})$ relations (3) have to be considered until the tree has $N_{\max}$ edges. As each of these relations can be obtained in an expected time which is polynomially bounded in $\log(q)$, we conclude that a tree with $N_{\max}$ edges can be constructed in an expected time of

$$\tilde{\mathcal{O}}(q^{2-\frac{2}{g}}).$$

Note that the depth of the tree is always bounded by s. In particular, as $s \in \mathcal{O}(\log(q))$, the depth of the tree is also in $\mathcal{O}(\log(q))$.

Step 4 – Relation generation. We now assume we have constructed a tree T with $N_{\max} = \lceil q^{1-\frac{1}{g}+\frac{1}{g^2}} \rceil$ edges.

Similarly to the above let

$$\begin{aligned} \mathcal{D} &:= \big\{ P_1 + \cdots + P_g \in \mathrm{Div}^g(\mathcal{C}) |\, \forall i = 1, \ldots, g : P_i \in \mathcal{F} \cup (T - \{*\}) \big\}, \\ \mathcal{D}^{ns} &:= \mathcal{D} \cap D^{g,ns}. \end{aligned}$$

Then $\mathcal{D}$ contains $\geq \frac{1}{g!} \cdot (\#\mathcal{F} + \#(T - \{*\}))^g \geq \frac{1}{g!} \cdot q^{g-1+\frac{1}{g}}$ elements. By the first two inequalities of (6), $\mathcal{D}^{ns}$ contains at least $\frac{3}{4g!} \cdot q^{g-1+\frac{1}{g}}$ elements. This means that the probability that the divisor D in relation (3) splits into elements of the factor base or vertices of the tree is

$$\geq \frac{3}{8g!} \cdot q^{-(1-\frac{1}{g})}.$$

The expected number of relations

$$[D] - \deg(D) \cdot [P_0] = \alpha a + \beta b \tag{8}$$

which have to be considered in each iteration of the For-loops is in $\mathcal{O}(q^{1-\frac{1}{g}})$. As each relation (8) can be obtained in an expected time which is polynomially bounded in $\log(q^g)$, this means that the expected running time of one iteration in the For-loops is in $\tilde{\mathcal{O}}(q^{1-\frac{1}{g}})$.

We have to generate $2(k+u) \cdot t \in \tilde{\mathcal{O}}(\#\mathcal{F}) = \tilde{\mathcal{O}}(q^{1-\frac{1}{g}})$ "combined" Full relations. This means that the total running time of this step of the algorithm is in $\tilde{\mathcal{O}}(q^{2-\frac{2}{g}})$.

Step 5 – Linear algebra. The linear algebra takes place on a sparse matrix with $\tilde{\mathcal{O}}(q^{1-1/g})$ rows and $\mathcal{O}(q^{1-1/g})$ columns.

As the tree has depth $\mathcal{O}(\log(q))$ and the size of the generating system is by assumption polynomially bounded in $\log(q)$, every row of the matrix $(R|S)$ contains only $\log(q)^{\mathcal{O}(1)}$ non-zero entries. As pointed out in the previous subsection, the computation can then be performed in an expected time of $\tilde{\mathcal{O}}(q^{2-2/g})$.

Final result. We have seen that Steps 1–5 of the algorithm all have an expected running time of $\tilde{\mathcal{O}}(q^{2-\frac{2}{g}})$. Moreover, by Proposition 3.4 and the properties of the linear algebra algorithm, after an expected number of $\mathcal{O}(\log\log(N)) \subseteq \mathcal{O}(\log(q))$ restarts of the computation of the matrix $(R|S)$, the linear algebra computation leads the solution to the discrete logarithm problem. This means that the total running time is in

$$\tilde{\mathcal{O}}(q^{2-\frac{2}{g}}),$$

in accordance with the statement in Proposition 3.1.

Storage requirements. Clearly, one can obtain: Only the first $\tilde{\mathcal{O}}(q^{1-\frac{1}{g}+\frac{1}{g^2}})$ registers are used and the bit-length of each number stored is polynomially bounded in $\log(q)$.

3.4. **On the number of special divisors.**

The purpose of this subsection is to prove Proposition 3.6.

We consider curves of a fixed genus g over finite fields.

Let $\mathcal{C}$ be such a curve over $\mathbb{F}_q$. Let $\mathrm{Div}^g(\mathcal{C})$ be the set of effective divisors of degree g on $\mathcal{C}$, and let E be a divisor of degree g on $\mathcal{C}$. We have the surjective map

$\mathrm{Div}^g(\mathcal{C}) \longrightarrow \mathrm{Cl}^0(\mathcal{C}), D \mapsto [D]-[E]$. Note that the set of special divisors of degree g is exactly the subset of $\mathrm{Div}^g(\mathcal{C})$ where the map to $\mathrm{Cl}^0(\mathcal{C})$ is not injective.

The number of special divisors is therefore bounded from above by $2\,(\#\,\mathrm{Div}^g(\mathcal{C}) - \#\,\mathrm{Cl}^0(\mathcal{C}))$, and it suffices to prove that $\#\,\mathrm{Div}^g(\mathcal{C}) - \#\,\mathrm{Cl}^0(\mathcal{C}) \in \mathcal{O}(q^{g-1})$.

We follow the exposition to the zeta-function in [Sti93]. Note, however, that we use different symbols for the indices.

Let $L = \prod_{i=1}^{2g}(1-\alpha_i t) \in \mathbb{C}(t)$ be the L-polynomial of $\mathcal{C}$, let A_n be the number of effective divisors of degree n, let B_n be the number of prime divisors of degree n on $\mathcal{C}$, and let[4]

$$S_n := \sum_{i=1}^{2g} \alpha_i^n .$$

As the α_i can be arranged such that $\alpha_i \alpha_{g+i} = q$ for all $i = 1, \ldots, g$, we have

$$\#\,\mathrm{Cl}^0(\mathcal{C}) = L(1) \in q^g - S_1 \cdot q^{g-1} + \mathcal{O}(q^{g-1}),$$

with $S_1 \in \mathcal{O}(\sqrt{q})$. We thus have to show that

$$A_g \in q^g - S_1 \cdot q^{g-1} + \mathcal{O}(q^{g-1}).$$

We will in fact show the more general statement

$$A_n \in q^n - S_1 \cdot q^{n-1} + \mathcal{O}(q^{n-1}) \tag{9}$$

for any fixed $n \in \mathbb{N}$.

Let us fix the following definition.

Definition 3.7. Let D be an effective divisor of degree n on $\mathcal{C}$ such that $D = \sum_{\ell=1}^n D_\ell$, where D_ℓ is a sum of e_ℓ prime divisors of degree ℓ. Then the vector $\underline{e} = (e_\ell)_\ell \in \mathbb{N}_0^n$ (with $\sum_\ell \ell e_\ell = n$) is called the *decomposition type* of D.

By sorting effective divisors of degree n by decomposition types, we obtain

$$A_n = \sum_{\underline{e}} \prod_{\ell} \binom{B_\ell + e_\ell - 1}{e_\ell}, \tag{10}$$

where the sum runs over all $\underline{e} \in \mathbb{N}_0^n$ with $\sum_\ell e_\ell \ell = n$ and the products run over $\ell \in \{1, \ldots, n\}$. We have

$$A_1 = B_1 = q + 1 - S_1,$$

which establishes the claim for $n = 1$. So let $n \geq 2$. By [Sti93, Proposition V.2.9], we have

$$B_\ell = \frac{1}{\ell} \cdot \sum_{m|\ell} \mu\left(\frac{\ell}{m}\right)(q^m - S_m) \in \frac{1}{\ell} \cdot q^\ell + \mathcal{O}(q^{\ell/2}) \subseteq \frac{1}{\ell} \cdot q^\ell + \mathcal{O}(q^{\ell-1}) \tag{11}$$

for $\ell \geq 2$.

This implies that

$$\begin{aligned} A_n &\in \sum_{\underline{e}} \frac{1}{e_1!}(q-S_1)^{e_1} \cdot \prod_{\ell \geq 2} \frac{1}{e_\ell!} \cdot \frac{1}{\ell^{e_\ell}} \cdot q^{\ell \cdot e_\ell} + \mathcal{O}(q^{n-1}) \\ &\subseteq \sum_{\underline{e}} \Big(\prod_{\ell} \frac{1}{e_\ell!} \cdot \frac{1}{\ell^{e_\ell}}\Big) \cdot (q^n - e_1 \cdot S_1 \cdot q^{n-1}) + \mathcal{O}(q^{n-1}). \end{aligned}$$

[4] The definition of S_n follows Equation (2.25) in [Sti93]. In [Sti93, Corollary V.1.17] an analogous definition is made with opposite sign.

In order to derive (9) it remains to show that

$$\sum_{\underline{e}} \prod_{\ell} \frac{1}{e_\ell !} \cdot \frac{1}{\ell^{e_\ell}} = 1 \tag{12}$$

and

$$\sum_{\underline{e}} e_1 \cdot \prod_{\ell} \frac{1}{e_\ell !} \cdot \frac{1}{\ell^{e_\ell}} = 1. \tag{13}$$

Equation (12) is equivalent to

$$\sum_{\underline{e}} n! \cdot \prod_{\ell} \frac{1}{e_\ell !} \cdot \frac{1}{\ell^{e_\ell}} = n!. \tag{14}$$

This is true because for any $\underline{e} \in \mathbb{N}_0^n$ with $\sum_\ell e_\ell \ell = n$, the set of permutations on n elements having exactly e_ℓ ℓ-cycles (for $\ell = 1, \ldots, n$) has $n! \cdot \prod_\ell \frac{1}{e_\ell !} \cdot \frac{1}{\ell^{e_\ell}}$ elements.

We come to equation (13). Note that we have a bijection

$$\begin{array}{ccc} \{\underline{e} \in \mathbb{N}_0^n \mid \sum_\ell e_\ell \ell = n,\ e_1 \neq 0\} & \longrightarrow & \{\underline{e}' \in \mathbb{N}_0^{n-1} \mid \sum_\ell e'_\ell \ell = n-1\}, \\ \underline{e} & \mapsto & \underline{e}' \end{array}$$

with $e'_1 = e_1 - 1$ and $e'_i = e_i$ for all $i = 1, \ldots, n-1$.

Equation (13) is then equivalent to

$$\sum_{\underline{e}'} \prod_{\ell} \frac{1}{e'_\ell !} \cdot \frac{1}{\ell^{e'_\ell}} = 1, \tag{15}$$

where the sum runs over all $\underline{e}' \in \mathbb{N}_0^{n-1}$ with $\sum_\ell e'_\ell \ell = n-1$ and the products run over $\ell \in \{1, \ldots, n-1\}$. We already know that this equation holds.

3.5. **Finding a generating system.** The main purpose of this subsection is to show Proposition 3.2. Note that in Proposition 3.2 we only consider curves of a fixed genus, represented by plane models of bounded degree. In this subsection, we consider arbitrary curves over finite fields. We show below how one can efficiently compute a small system of degree 0 divisor classes which with a probability $\geq \frac{1}{2}$ generates the degree 0 class group, provided the L-polynomial is known (see Proposition 3.17). Proposition 3.2 follows from this statement and the fact—already mentioned in subsection 3.2—that one can then compute the L-polynomial in a time which is polynomially bounded in $\log(q)$ ([Pil90], [Pil91]). On our way to prove Proposition 3.17, we also show how one can efficiently compute uniformly randomly distributed effective divisors of a specific degree, provided one knows the L-polynomial, a result which might be of independent interest (see Proposition 3.16).

As usual, curves are represented by plane models, and divisors are given in ideal representation, as detailed in Section 2.

Proposition 3.8. *Given a curve $\mathcal{C}$ over a finite field $\mathbb{F}_q$, represented by a plane model of degree d, and a natural number n, one can with a randomized algorithm*

- *decide if $\mathcal{C}$ has an $\mathbb{F}_q$-rational point;*
- *if this is the case, compute such a point which is uniformly randomly distributed in $\mathcal{C}(\mathbb{F}_q)$*

in an expected time which is polynomially bounded in $d \cdot \log(q)$.

Proof. We still assume that the covering $x : \mathcal{C} \longrightarrow \mathbb{P}^1_{\mathbb{F}_q}$ is separable, and as usual we set $r := \deg(x)$.

By the Hasse-Weil bound, we have $\#\mathcal{C}(\mathbb{F}_q) \geq q + 1 - 2gq^{1/2}$. This means that for $q \geq 4g^2$, $\mathcal{C}(\mathbb{F}_q)$ is non-empty. The algorithm depends on a case distinction:

If $q < d^4$ we compute a list of the elements in $\mathcal{C}(\mathbb{F}_q)$ by iterating over all elements P of $\mathbb{P}^1(\mathbb{F}_q)$ and computing for each such P the divisor $x^{-1}(P)$ in free representation. If it turns out that $\mathcal{C}(\mathbb{F}_q)$ is empty, we output that this is the case, otherwise we choose one of the points in $\mathcal{C}(\mathbb{F}_q)$ uniformly at random. We have already argued in subsection 3.3 that this computation can be performed in an expected time which is polynomially bounded in $d \cdot q$, that is, in an expected time which is polynomially bounded in d as q is also polynomially bounded in d by assumption.

If $q \geq d^4$ (such that $q > 4g^2$ and therefore $\mathcal{C}(\mathbb{F}) \neq \emptyset$), we proceed with the following algorithm.

Algorithm for computation of a uniformly randomly distributed rational point on a curve over a finite field.

Input: A curve $\mathcal{C}/\mathbb{F}_q$, represented by a plane model, with $\mathcal{C}(\mathbb{F}_q) \neq \emptyset$.

```
1. Choose a point $P \in \mathbb{P}^1(\mathbb{F}_q)$ uniformly at random.
2. Compute $x^{-1}(P)$ in free representation.
3. Let $P_1, \ldots, P_a$ be the distinct $\mathbb{F}_q$-rational points occurring in $x^{-1}(P)$.
      (The other prime divisors in the support of $x^{-1}(P)$ are ignored, and $a = 0$ is
   possible.)
4. Choose a number $i$ in $\{1, \ldots, r\}$ uniformly at random.
5. If $i \leq a$, output $P_i$, otherwise go back to Step 1.
```

Without any assumption on q and g, this algorithm computes a uniformly randomly distributed point in $\mathcal{C}(\mathbb{F}_q)$ provided that this set is non-empty, as we show now.

The computation of Steps 1–5 can be performed in an expected time which is polynomially bounded in $d \cdot \log(q)$.

Let us analyze the algorithm: After Step 4 the following always holds: The random variable (P, i) is uniformly distributed on the set $\mathbb{P}^1(\mathbb{F}_q) \times \{1, \ldots, r\}$, which has $(q + 1) \cdot r$ elements. This means that in every iteration of the algorithm, the probability that the algorithm terminates in Step 5 is always $\frac{\#\mathcal{C}(\mathbb{F}_q)}{(q+1)\cdot r}$, and if this is the case, every point in $\mathcal{C}(\mathbb{F}_q)$ is chosen with the same probability of $\frac{1}{\#\mathcal{C}(\mathbb{F}_q)}$.

It follows that the output of the algorithm is a uniformly randomly distributed element in $\mathcal{C}(\mathbb{F}_q)$. Moreover, the expected number of iterations is $\frac{(q+1)\cdot r}{\#\mathcal{C}(\mathbb{F}_q)}$. In order to prove the proposition, we therefore have to show that the quantity $\frac{q+1}{\#\mathcal{C}(\mathbb{F}_q)}$ is polynomially bounded in $d \cdot \log(q)$; we show that it is polynomially bounded in g (which in turn is polynomially bounded in d).

We have $\#\mathcal{C}(\mathbb{F}_q) \geq q + 1 - 2gq^{1/2}$ by the Hasse-Weil bound. For $q \geq 16g^2$ we have $\#\mathcal{C}(\mathbb{F}_q) \geq \frac{q}{2} + 1$, and therefore $\frac{q+1}{\#\mathcal{C}(\mathbb{F}_q)} \leq 2$. On the other hand, for $q < 16g^2$ the quantity $\frac{q+1}{\#\mathcal{C}(\mathbb{F}_q)}$ is clearly polynomially bounded in g. □

Proposition 3.9. *Given a curve $\mathcal{C}$ over a finite field $\mathbb{F}_q$, represented by a plane model of degree d and a natural number n, one can with a randomized algorithm*

- *decide if $\mathcal{C}$ has a prime divisor of degree n*

- *if this is the case, compute such a prime divisor which is uniformly randomly distributed in the set of all prime divisors of $\mathcal{C}$ of degree n*

in an expected time which is polynomially bounded in $d \cdot n \cdot \log(q)$.

Proof. As in subsection 3.4, let B_n be the number of prime divisors of $\mathcal{C}$ of degree n. Then we have (cf. [Sti93, Corollary V.2.10.]):

$$B_n \geq \frac{q^n}{n} - (2+7g) \cdot \frac{q^{n/2}}{n}.$$

For $q > (2+7g)^{2/n}$ we therefore have $B_n \geq 1$.

Similarly to the algorithm for the previous proposition, we have a case distinction according to $q > (2+7d^2)^2$.

In both cases we consider $\mathbb{F}_{q^n}$-rational points of $\mathcal{C}_{\mathbb{F}_{q^n}}$. Note that given such a point, one can compute the associated prime divisor (=closed point) of $\mathcal{C}$ by computing the intersection of the corresponding prime ideal with the function field $\mathbb{F}_q(\mathcal{C})$ in a time which is polynomially bounded in $d \cdot n \cdot \log(q)$ (cf. Section 2).

If $q \leq (2+7d^2)^2$, we iterate over all $\mathbb{F}_{q^n}$-rational points of $\mathcal{C}_{\mathbb{F}_{q^n}}$ (as described at the beginning of the proof of Proposition 3.8). For each such point we compute the associated prime divisor (=closed point) of $\mathcal{C}$, and check if this is a prime divisor of degree n. In this way we check if a prime divisor of degree n on $\mathcal{C}$ exists, and if this is the case, we uniformly randomly choose one.

So now let $q > (2+7d^2)^2$. Then, in particular, $q > (2+7d^2)^{2/n}$, and therefore there is a prime divisor of degree n on $\mathcal{C}$. Then the algorithm is also easy: We first choose an $\mathbb{F}_{q^n}$-rational point of $\mathcal{C}_{\mathbb{F}_{q^n}}$ uniformly at random compute the associated prime divisor (=closed point) of $\mathcal{C}$. If the prime divisor has degree n, we output it, otherwise we repeat this process.

Here we compute the $\mathbb{F}_{q^n}$-rational point using an algorithm for Proposition 3.8, adapted to work over extension fields. The expected time of one iteration is polynomially bounded in $d \cdot n \cdot \log(q)$. We therefore have to show that the number of iterations is polynomially bounded in $d \cdot n \cdot \log(q)$.

The number of points in $\mathcal{C}(\mathbb{F}_{q^n})$ such that the associated prime divisor of $\mathcal{C}$ has degree n is $n \cdot B_n$. Therefore the probability that a uniformly distributed point in $\mathcal{C}(\mathbb{F}_{q^n})$ does *not* give rise to a prime divisor of degree n is $\frac{\#\mathcal{C}(\mathbb{F}_{q^n}) - nB_n}{\#\mathcal{C}(\mathbb{F}_{q^n})}$ which is $\leq \frac{(2+9g)\cdot q^{n/2}}{q^n - 2gq^{n/2}} = \frac{(2+9g)}{q^{n/2}-2g}$. For $q \geq (4+20g)^{2/n}$ this is $\leq \frac{1}{2}$. If this bound in not satisfied, the probability that a uniformly distributed point in $\mathcal{C}(\mathbb{F}_{q^n})$ does give rise to a prime divisor of degree n is in $\frac{1}{g^{\Omega(1)}}$. □

We aim at giving an efficient algorithm to compute a random effective divisor which is uniformly randomly distributed in the set of all effective divisors of a specific degree. We give some definitions and notations first for which we fix a curve $\mathcal{C}$ over a finite field.

Definition 3.10. Let $n \in \mathbb{N}_0$ and $m \in \mathbb{N}$. Then, as usual, an effective divisor on $\mathcal{C}$ of degree n which is the sum of prime divisors of degree at most m is called *m-smooth* or *(n,m)-smooth*. An effective divisor on $\mathcal{C}$ of degree n which is the sum of prime divisors of degree exactly m is called *m-homogeneous* or *(n,m)-homogeneous.*

Notation 3.11. Let $n \in \mathbb{N}_0$ and $m \in \mathbb{N}$. Then the number of effective divisors on $\mathcal{C}$ of degree n which split into prime divisors of degree $\leq m$ (resp. $= m$, resp. $\geq m$) is denoted by $\psi_{\leq}(n,m)$ (resp. $\psi_{=}(n,m)$, resp. $\psi_{\geq}(n,m)$).

Note that in the notation of subsection 3.4,

$$\psi_{\leq}(n,n) = A_n \text{ and } \psi_{=}(n,n) = B_n.$$

Definition 3.12. Let D be an $(r \cdot m, m)$-homogeneous divisor on $\mathcal{C}$. Then the *multiplicity type* $\underline{\mu} \in \mathbb{N}_0^r$ of D is defined as follows: μ_i is the number of prime divisors occurring with multiplicity exactly i in D.

Note that $\sum_{i=1}^r i\mu_i = r$.

Proposition 3.13. *There exists a randomized algorithm such that the following holds: Given a natural number m, a curve $\mathcal{C}$ over a finite field $\mathbb{F}_q$, represented by a plane model of degree d, such that $\mathcal{C}$ has at least one prime divisor of degree m, the number B_m of prime divisors of degree m on $\mathcal{C}$ as well as another natural number r, the algorithm computes an effective divisor on $\mathcal{C}$ which is uniformly randomly distributed among all $(r \cdot m, m)$-homogeneous effective divisors. Moreover, the expected running time is polynomially bounded in $m \cdot r \cdot d \cdot \log(q)$.*

Proof. Before we come to the proof, we would like to caution the reader that it is incorrect to merely choose r prime divisors of degree m uniformly at random; this would not lead to a uniformly distributed output.

Let an instance as in the proposition be given.

An outline of the algorithm is as follows: First we compute a random vector $\underline{\mu} \in \mathbb{N}_0^r$ whose distribution is equal to the distribution of the multiplicity type of an effective divisor which is uniformly randomly distributed among all $(r \cdot m, m)$-homogeneous divisors. Then we choose prime divisors of degree m $P_{i,j}$ with $i = 1, \ldots, r$ and $j = 1, \ldots, \mu_i$, without repetition such that the tuple $(P_{i,j})_{i,j}$ is uniformly distributed among all such tuples. We output the divisor $\sum_{i=1}^r i \cdot \sum_{j=1}^{\mu_i} P_{i,j}$. The output is then clearly uniformly distributed among all $(r \cdot m, m)$-homogeneous divisors.

As for the details, we compute the $P_{i,j}$ iteratively, discarding duplicates. The expected running time of this part of the computation is then as claimed. It remains to be shown that the vector $\underline{\mu}$ can be computed in the desired expected time.

The notion of multiplicity type immediately generalizes to arbitrary free abelian monoids, where previously we considered the free abelian monoid generated by prime divisors of degree m on $\mathcal{C}$. And a free abelian monoid on a finite set P is the same as the set of multisets whose elements are in P (only the notation is different). As above, let B_m be the number of prime divisors of degree m on $\mathcal{C}$. Then the multiplicity type of a multiset which is uniformly randomly distributed among all multisets of cardinality r and elements from $\{1, \ldots, B_m\}$ has the desired distribution. Now, the multisets of cardinality r with elements in $\{1, \ldots, B_m\}$ are in canonical bijection with the tuples $(x_1, \ldots, x_r) \in \{1, \ldots, B_m\}^r$ with $x_i \leq x_j$ for $i < j$. These tuples are—as is well known—in bijection with the tuples $(x_1', \ldots, x_r') \in \{1, \ldots, B_m + r - 1\}$ with $x_i' < x_j'$ (via $x_i' = x_i + i - 1$), and these are in bijection with the subsets of cardinality r of $\{1, \ldots, B_m + r - 1\}$.

Note that B_m is part of the input. So, to compute the random vector $\underline{\mu}$, we can proceed as follows: We choose a subset of cardinality r from $\{1, \ldots, B_m + r - 1\}$ uniformly at random. From this set, we compute the corresponding tuple $(x_1, \ldots, x_r)$ and then the multiplicity type of the corresponding multiset. Clearly, all this can be done in an expected time of $(r \cdot \log(B_m))^{\mathcal{O}(1)} \subseteq (r \cdot m \cdot \log(q))^{\mathcal{O}(1)}$. □

Lemma 3.14. *Given the L-polynomial of a curve of genus g over a finite field $\mathbb{F}_q$ and two natural numbers $m \leq n$, one can compute the numbers $\psi_{\leq}(n,m), \psi_{=}(n,m)$ and $\psi_{\geq}(n,m)$ in a time which is polynomially bounded in $n \cdot g \cdot \log(q)$.*

The *proof* of this lemma is inspired by the product formula for the zeta-function. The zeta-function is

$$\sum_{i\in\mathbb{N}} A_i t^i = \sum_{D \text{ an eff. divisor}} t^{\deg D}$$

$$= \prod_{P \text{ a prime divisor}} (1 - t^{\deg(P)})^{-1} = \prod_{\ell\in\mathbb{N}} (1-t^\ell)^{-B_\ell}.$$

Similarly, for all $m \in \mathbb{N}$,

$$\sum_{i\in\mathbb{N}} \psi_{\leq}(i,m) t^i = \prod_{\ell \leq m} (1-t^\ell)^{-B_\ell}, \tag{16}$$

$$\sum_{i\in\mathbb{N}} \psi_{=}(i,m) t^i = (1-t^m)^{-B_m}, \tag{17}$$

and

$$\sum_{i\in\mathbb{N}} \psi_{\geq}(i,m) t^i = \prod_{\ell \geq m} (1-t^\ell)^{-B_\ell}. \tag{18}$$

The algorithm to compute $\psi_{\leq}(n,m)$ is as follows:

Let the L-polynomial $L(t)$ be given. We first compute $S_1, \ldots, S_m$ via Newton's identities (that is, via the equation $L'(t) = -L(t) \cdot (\sum_{i=1}^{\infty} S_i t^{i-1})$) from the coefficients of the L-polynomial. From these we compute $B_1, \ldots, B_m$ using (11). Then we compute $\prod_{\ell \leq m} (1-t^\ell)^{B_\ell}$ and the inverse of its residue class modulo t^{n+1}, which is $\sum_{i=1}^{n} \psi_{\leq}(i,m)\,[t]^i_{(t^{n+1})}$.

The other algorithms operate similarly. □

Lemma 3.15. *Given the L-polynomial of a curve g over a finite field $\mathbb{F}_q$ and natural numbers n, m with $m \leq n$, one can with a randomized algorithm compute a random tuple $\underline{e} \in \mathbb{N}_0^n$ with $\sum_\ell \ell e_\ell = n$ whose distribution is equal to the distribution of the decomposition type of a random effective divisor on $\mathcal{C}$ which is uniformly distributed among all (n,m)-smooth divisors in an expected time which is polynomially bounded in $g \cdot n \cdot \log(q)$.*

Proof. The algorithm operates in a recursive way and is based on partitioning the set of (n,m)-smooth divisors into subsets, according to how many prime divisors (with multiplicities) of degree m occur in an (n,m)-smooth divisor.

If $m \geq 2$ and $\ell \in \{0, \ldots, \lfloor \frac{n}{m} \rfloor\}$, there are

$$\psi_{=}(\ell m, m) \cdot \psi_{\leq}(n - \ell m, m-1)$$

(n,m)-smooth divisors of the form $\sum_{i=1}^{\ell} P_i + D'$, where the P_i are prime divisors of degree m and D' is an $m-1$-smooth divisor.

Algorithm for computation of a random tuple reflecting the distribution type of a random smooth divisor.

Input: L, the L-polynomial of a curve over a finite field and two natural numbers n, m. (The algorithm is called by $A(L,n,m)$).

If $m = 1$, output $(n, 0, \ldots, 0) \in \mathbb{N}_0^n$. Otherwise:

1. Compute $\psi_{\leq}(n,m)$ and the numbers $a_\ell \longleftarrow \psi_{=}(\ell m, m) \cdot \psi_{\leq}(n-\ell m, m-1)$ for $\ell = 0, \ldots, \lfloor \frac{n}{m} \rfloor$.
2. Let $b_\ell \longleftarrow \sum_{i=0}^{\ell} a_i$ for $\ell = 0, \ldots, \lfloor \frac{n}{m} \rfloor$; let $b_{-1} \longleftarrow 0$.
3. Choose a natural number $x \leq \psi_{\leq}(n,m)$ uniformly at random.
4. Determine ℓ such that $x \in [b_{\ell-1}+1, b_\ell]$.
5. Output
$$(0,\ldots,0,\ell,0,\ldots,0) + (A(L, n-\ell m, m-1) \,|\, \underline{0}) \in \mathbb{N}_0^n,$$
where the non-trivial entry in the first tuple is at index m and $(A(L, n-\ell m, m-1) \,|\, \underline{0})$ is the concatenation of the output of the algorithm applied to $L, n-\ell m, m-1$ and the zero-tuple of length ℓm.

By the remarks above the algorithm, the distribution of the random variable ℓ in Step 4 of the algorithm is equal to the distribution of the number of (n,m)-smooth divisors being the sum of ℓ prime divisors of degree m and an $m-1$-smooth divisor.

It follows by induction on m that the algorithm operates correctly. Moreover, a running time as claimed can be achieved by Lemma 3.14. □

We now easily obtain:

Proposition 3.16. *Given a curve $\mathcal{C}$ over a finite field $\mathbb{F}_q$, represented by a plane model of degree d, as well as its L-polynomial, and a natural number n, one can with a randomized algorithm*

- *decide if there is an effective divisor of degree n on $\mathcal{C}$;*
- *compute a random effective divisor of degree n which is uniformly randomly distributed in the set of all effective divisors of degree n on $\mathcal{C}$*

in an expected time which is polynomially bounded in $d \cdot n \cdot \log(q)$.

Proof. The first statement is obvious because the L-polynomial is given.

Given the previous results, the algorithm for this proposition is straightforward: We compute a random tuple $\underline{e} \in \mathbb{N}_0^n$ with $\sum_\ell e_\ell \ell = n$ whose distribution is equal to the distribution of the decomposition type of a random effective divisor on $\mathcal{C}$ which is uniformly distributed among all effective divisors of degree n.

Then for each $\ell = 1, \ldots, n$, we compute a uniformly randomly distributed $(e_\ell \ell, \ell)$-homogeneous effective divisor D_ℓ.

We output the divisor $\sum_\ell D_\ell$.

By Proposition 3.13 and Lemma 3.15 these computations can be performed in the claimed expected running time. □ □

Proposition 3.17. *Given a curve $\mathcal{C}$ over a finite field $\mathbb{F}_q$, represented by a plane model of degree d as well as its L-polynomial and a divisor D_0 of degree 1 whose height is polynomially bounded in $d \cdot \log(q)$, one can with a randomized algorithm compute a uniformly randomly distributed element of $\mathrm{Cl}^0(\mathcal{C})$, represented by an along D_0 reduced divisor, in an expected time which is polynomially bounded in $d \cdot \log(q)$.*

Proof. Every divisor D of degree $\geq 2g-1$ is non-special and thus the linear system $|D|$ is isomorphic to $\mathbb{P}^{\deg(D)-g}(\mathbb{F}_q)$.

Therefore, if D is an effective divisor which is uniformly distributed among all divisors of some degree $n \geq 2g-1$, $[D - \deg(D) \cdot D_0]$ is uniformly distributed in $\mathrm{Cl}^0(\mathcal{C})$. Moreover, $2g-1 \leq (d-1)(d-2)-1$.

In order to compute the desired uniformly distributed divisor class, we first compute a uniformly distributed effective divisor D of degree $(d-1)(d-2)-1$, and then we compute its reduction along D_0.

The computations can be performed in the claimed expected running time by the previous proposition and remarks in Section 2. □

We now prove:

Proposition 3.18. *There exists a randomized algorithm such that the following holds: Given a curve $\mathcal{C}$ over a finite field $\mathbb{F}_q$, represented by a plane model of degree d, as well as its L-polynomial and a divisor D_0 of degree 1 whose height is polynomially bounded in $d \cdot \log(q)$, the algorithm computes a system of elements $c_1, \ldots, c_u \in \mathrm{Cl}^0(\mathcal{C})$, represented by along D_0 reduced divisors, where $u := e\ell$ with $e := \lceil \log_2(\# \mathrm{Cl}^0(\mathcal{C})) \rceil$ and $\ell := \lceil \log_2(e)+1 \rceil$. Moreover, the expected running time is polynomially bounded in $d \cdot \log(q)$, and with a probability $\geq \frac{1}{2}$, the system $c_1, \ldots, c_u$ is a generating system of $\mathrm{Cl}^0(\mathcal{C})$.*

The proposition immediately follows from the preceding proposition and the following lemma (cf. the proof of [Heß05, Lemma 50]).

Lemma 3.19. *Let G be a finite abelian group with N elements, let $e := \lceil \log_2(N) \rceil$, $\ell := \lceil \log_2(e) + 1 \rceil, u := e\ell$, and let $g_1, \ldots, g_u$ be uniformly distributed random elements of G. Then with a probability of $\geq \frac{1}{2}$, $g_1, \ldots, g_u$ generate G.*

Proof. To begin with, let H be a proper subgroup of G, and let $g_1, \ldots, g_a$ be uniformly randomly distributed elements from G. Then as $\frac{\#H}{\#G} \leq \frac{1}{2}$, with a probability $\leq \frac{1}{2^a}$, all g_i lie in H, that is, with a probability $\geq 1 - \frac{1}{2^a}$, $H \subsetneq \langle H, g_1, \ldots, g_a \rangle$.

It follows by induction on b: Let $a, b \in \mathbb{N}$, and let $g_1, \ldots, g_{ab}$ be uniformly randomly distributed elements from G. Then with a probability $\geq (1 - \frac{1}{2^a})^b \geq 1 - \frac{b}{2^a}$, $\langle g_1, \ldots, g_a \rangle$ contains at least $\min\{N, 2^b\}$ elements.

With $b := e$ and $a := \ell$, we have $2^b \geq N$ and $\frac{b}{2^a} \leq \frac{1}{2}$. The lemma thus follows. □

Proposition 3.2 follows from Proposition 3.18 and the fact—already mentioned in subsection 3.2—that for curves of a bounded degree over finite fields, one can compute the L-polynomial in polynomially bounded time in $\log(q)$, where $\mathbb{F}_q$ is the ground field.

4. Index calculus for curves of lower-bounded genus

4.1. **Overview.** In order to establish Theorem 2, we combine algorithms for Theorem 1 with an algorithm for "sufficiently high genus curves". For this second algorithm, we make use of some results from [Heß05], in particular, a certain result on smoothness probabilities. We note that the result we need is merely an easy corollary of a result in [Heß05] which contains much more information if the genus of the curve is in a certain sense large against the bit-length of the cardinality of the ground field.

We give an outline of the proof of Theorem 2. Below we prove the following result.

Proposition 4.1. *Let $C > 0$ be fixed. Then there exists some $g_1 \in \mathbb{N}$ such that one can solve the discrete logarithm problem in the degree 0 class groups of curves* *of genus* $\geq g_1$ *in an expected time of* $\mathcal{O}((q^g)^C)$*, where* q *is the cardinality of the ground field and* g *is the genus.*

We show now how one can obtain Theorem 2 via this result and Theorem 1. We first consider the second statement in Theorem 2.

Let g_0 be a natural number ≥ 2, and let $C := \frac{2}{g_0}(1 - \frac{1}{g_0})$. Let g_1 be as in the proposition for this constant C.

Then given an instance of the discrete logarithm problem with a curve of genus $g \geq g_0$, we first compute its genus g. Then we proceed with a case distinction. If the genus is $\geq g_1$ we apply an algorithm which satisfies Proposition 4.1. If the genus is $< g_1$, we apply an algorithm which satisfies Theorem 1 for genus g curves.

We now come to the first statement in Theorem 2. By the lower bound on $\# \mathrm{Cl}^0(\mathcal{C})$ from [LMD90] stated in Section 2 (equation (2)), q^g is in $\tilde{\mathcal{O}}(\# \mathrm{Cl}^0(\mathcal{C}))$. The second statement in Theorem 2 therefore immediately also gives the first statement.

4.2. **The algorithm.** We now outline an algorithm for Proposition 4.1, accompanied by statements on running times. The algorithm is substantially easier to state than the index calculus algorithm in Section 3: It is a "basic" index calculus algorithm without large prime variation. The algorithm is closely related to the algorithms in [Heß05], and we make use of various results from [Heß05].

For the index calculus algorithm we again need a generating system. We could proceed as in Section 3: Using a very small system which, with probability $\geq \frac{1}{2}$, is a generating system and terminating the algorithm if a predefined time bound is reached. However, in the situation we are concerned with here, there is an easier solution: It is shown in [Heß05] that one can compute a generating system of a size which is polynomially bounded in $q \cdot g$ in an expected time which is also polynomially bounded in $q \cdot g$ (see [Heß05, Theorem 34 and Algorithm 35]).

Note that generally if any subroutine of our algorithm has an expected running time which is polynomially bounded in $q \cdot g$, then this subroutine in particular has an expected running time of $\mathcal{O}(q^{Cg})$ for g large enough. Thus if such a subroutine is executed once or even a number of times which is polynomially bounded in $q \cdot g$, then the running time of this subroutine is not critical for the establishment of the desired result.

In the algorithm, we again first compute the group order N and factor it. We fix an appropriate smoothness bound S and define the factor base as the set of prime divisors on $\mathcal{C}$ of degree $\leq S$. After this, we generate a relation matrix. Here we follow Step 4 of the algorithm for Theorem 1; generating the relations in a different way however. After this, we proceed as in the algorithm for Theorem 1.

We now describe the important steps of the algorithm.

Obviously, we only have to establish Proposition 4.1 for constants C of the form $C = \frac{1}{c}$ for $c \in \mathbb{N}$. So let $c \in \mathbb{N}$ be fixed, and let $C := \frac{1}{c}$.

As usual, let us fix an instance consisting of a curve $\mathcal{C}/\mathbb{F}_q$, a divisor D_0 of degree $1, a, b \in \mathrm{Cl}^0(\mathcal{C})$ with $b \in \langle a \rangle$, and a generating system $c_1, \ldots, c_u$.

Computation of the group order and factorization. A. Lauder and D. Wan have shown in [LW08] that one compute the order of the degree 0 class group of a curve over $\mathbb{F}_q$ given by a plane model of degree d in a time which is polynomially bounded in $q \cdot d$. (In fact, if $q = p^e$ with p prime and $n \in \mathbb{N}$, the running time is polynomially bounded in $p \cdot e \cdot d$.) Note that as by our assumption $d \in \mathcal{O}(g)$, the running time is, in particular, polynomially bounded in $q \cdot g$.

The order can be factored in subexponential time in $\# \operatorname{Cl}^0(\mathcal{C})$ (cf. [Pom87]). This implies that it can also be factored in subexponential time in q^g. Therefore this computation is not time critical.

Construction of the factor base. We first compute the genus g. We fix a "smoothness bound" $m := \lceil \frac{g}{8c} \rceil$, and let the factor base $\mathcal{F}$ be the set of prime divisors of degree $\leq m$. Here we construct the set of prime divisors by iterating over all prime divisors of degree $\leq m$ on $\mathbb{P}^1_{\mathbb{F}_q}$ and considering the preimages under the covering $x : \mathcal{C} \longrightarrow \mathbb{P}^1_{\mathbb{F}_q}$.

The factor base has $\leq r \cdot (q+1)^m$ elements, where $r := \deg(x)$. For $g \geq 8c$, we have $m \leq \frac{g}{4c}$. Since $r \leq d \in \mathcal{O}(g)$, the size of the factor base is then (for $g \geq 8c$) in $\tilde{\mathcal{O}}(q^{\frac{g}{4c}})$, and the expected running time is also in $\tilde{\mathcal{O}}(q^{\frac{g}{4c}})$.

Relation generation. Let $t \in \mathbb{N}$ be defined as in the algorithm for Theorem 1. Now as in the algorithm for Theorem 1, we generate relations in two different ways: First we generate $(k+2u) \cdot t$ relations as follows: We choose $\alpha, \beta, s_1, \ldots, s_u \in \mathbb{Z}/N\mathbb{Z}$ uniformly at random and choose uniformly randomly a divisor D in the class $s_1 c_1 + \cdots + s_u c_u + \alpha a + \beta b + (2g-1) \cdot [D_0]$. Second for $i_1 = 1, \ldots, k$, we generate t relations by again choosing $\alpha, \beta, s_1, \ldots, s_u \in \mathbb{Z}/N\mathbb{Z}$ uniformly at random and choosing uniformly randomly a divisor D in the class $[F_{i_1}] - [D_0] + s_1 c_1 + \cdots + s_u c_u + \alpha a + \beta b + (2g-1) \cdot [D_0]$. In any case, if D is m-smooth, we have obtained a relation between input elements and factor base elements.

Each iteration can be performed in an expected time of $(g \log(q))^{\mathcal{O}(1)} \cdot u$. Note also that as the random divisor class $s_1 c_1 + \cdots + s_u c_u + \alpha a + \beta b + (2g-1)[D_0]$ is uniformly distributed in the set of divisor classes of degree $2g-1$ and divisors of degree $\geq 2g-1$ are non-special, the random divisor D is uniformly distributed in the set of all effective divisors of degree $2g-1$.

In order to bound the expected running time for the generation of one relation between input elements and factor base elements, we need a lower bound on the probability that a uniformly distributed random divisor of degree $2g-1$ is m-smooth. For this we can use [Heß05, Theorem 8] which gives a much more precise statement than the one we need. In fact, just from the fact that $m \in \Omega(2g-1)$, we learn from [Heß05, Theorem 8] that the probability in question is in $\frac{1}{g^{\Omega(1)}}$.

This establishes that the expected running time needed to generate one relation is in $(g \cdot \log(q))^{\mathcal{O}(1)} \cdot u$, and the total expected time to generate the relation matrix is therefore in $(g \cdot \log(q))^{\mathcal{O}(1)} \cdot u \cdot \tilde{\mathcal{O}}(\#\mathcal{F})$. For $g \geq 8c$ (such that the size of the factor base is in $\tilde{\mathcal{O}}(q^{\frac{g}{4c}})$) this is contained in $(g \cdot \log(q))^{\mathcal{O}(1)} \cdot u \cdot \tilde{\mathcal{O}}(q^{\frac{g}{4c}})$.

For g large enough, the expected time for the construction of the relation matrix is then in $\tilde{\mathcal{O}}(q^{\frac{g}{2c}})$, as is the number of non-trivial entries.

Linear algebra. We use an algorithm from sparse linear algebra, as in the previous algorithm. For g large enough the linear algebra then takes an expected time of $\tilde{\mathcal{O}}(q^{\frac{3g}{4c}})$

Final result. We see that for g large enough, the expected running time is in $\mathcal{O}(q^{\frac{g}{c}})$.

5. Index calculus for elliptic curves over extension fields of a fixed degree

5.1. **Overview.** In this section we establish Theorem 3. The algorithm is closely related to the algorithm for Theorem 1, and the structures of the algorithms are the same. The key differences concern the definition of the factor base and the relation generation.

In subsection 5.3 we give an index calculus algorithm with double large prime variation which leads to the following proposition. We alert the reader to the similarity with Proposition 3.1.

Proposition 5.1. *Let us fix some $n \geq 2$. Then there exists a randomized algorithm such that the following holds: Upon input of an elliptic curve $E/\mathbb{F}_{q^n}$, elements $A, B \in E(\mathbb{F}_{q^n})$ with $B \in \langle A \rangle$ and a system $C_1, \ldots, C_u$ of elements of $E(\mathbb{F}_{q^n})$, if the algorithm terminates, it outputs the discrete logarithm of B with respect to A. Moreover, if $C_1, \ldots, C_u$ forms a generating system of $E(\mathbb{F}_{q^n})$ and u is polynomially bounded in $\log(q)$, the expected running time of the algorithm is in $\tilde{\mathcal{O}}(q^{2-\frac{2}{n}})$.*

The algorithm uses only the first $\tilde{\mathcal{O}}(q^{1-\frac{1}{n}+\frac{1}{n^2}})$ registers, and each register always contains elements whose bit-length is polynomially bounded in $\log(q)$.

Together with the arguments in Section 3 this proposition establishes Theorem 3.

The algorithm for this proposition is also quite similar to the algorithm for Proposition 3.1. The only essential difference is that we define the factor base and generate the relations is a different way.

Briefly, the factor base is defined as follows: We fix a covering $\varphi : E \longrightarrow \mathbb{P}^1_{\mathbb{F}_q}$ of degree 2 satisfying a certain condition which is stated below (Condition 5.8). Then the factor base is some subset of $\{P \in E(\mathbb{F}_{q^n}) \mid \varphi(P) \in \mathbb{P}^1(\mathbb{F}_q)\}$ consisting of $\lceil q^{1-\frac{1}{n}} \rceil$ or $\lceil q^{1-\frac{1}{n}} \rceil + 1$ elements.

Relations are generated by what we call a *decomposition algorithm.* This algorithm relies on solving systems of multivariate polynomial equations over $\mathbb{F}_q$. The systems are derived from so-called summation polynomials.

In the next subsection we give some background information and specify what exactly we mean by a decomposition algorithm, following [Die10]. We also give the key result for the analysis of the algorithm in the present situation. More information and proofs can be found in [Die10]. Then in subsection 5.3 we discuss the index calculus algorithm.

5.2. **The decomposition algorithm.** The key properties on the summation polynomials we need are stated in the following two propositions.

Proposition 5.2. *Let E be an elliptic curve over a field k, and let us fix a covering $\varphi : E \longrightarrow \mathbb{P}^1_k$ of degree 2 with $\varphi \circ [-1] = \varphi$. Let $m \in \mathbb{N}$ with $m \geq 2$. Then there exists an up to multiplication by a non-trivial constant unique irreducible multihomogeneous polynomial $S_{\varphi,m} \in k[\mathrm{X}_1, \mathrm{Y}_1, \mathrm{X}_2, \mathrm{Y}_2, \ldots, \mathrm{X}_m, \mathrm{Y}_m]$ such that for all $P_1, \ldots, P_m \in E(\overline{k})$ we have $S_{\varphi,m}(\varphi(P_1), \ldots, \varphi(P_m)) = 0 \longleftrightarrow \exists \epsilon_1, \ldots, \epsilon_m \in \{1, -1\}$ such that $\epsilon_1 P_1 + \cdots \epsilon_m P_m = O$. The polynomial $S_{\varphi,m}$ has multidegree $(2^{m-2}, \ldots, 2^{m-2})$.*

Definition 5.3. We call a *multihomogeneous* polynomial $S_{\varphi,m}$ as in the proposition an mth *summation polynomial of E with respect to φ.*

Remark 5.4. The summation polynomials introduced here are the "homogenized variants" of the summation polynomials in [Sem04].

Proposition 5.5. *Given an elliptic curve in Weierstraß form over a finite field $\mathbb{F}_q$ $m \in \mathbb{N}$ with $m \geq 2$ and $\varphi : E \longrightarrow \mathbb{P}^1_k$ of degree 2 with $\varphi \circ [-1] = \varphi$, the m^{th} summation polynomial with respect to the covering $\varphi : E \longrightarrow \mathbb{P}^1_{\mathbb{F}_q}$ can be computed with a randomized algorithm in an expected time which is polynomially bounded in $e^{m^2} \cdot \log(q)$.*

Note that for fixed m the expected running time is polynomially bounded in $\log(q)$.

These two propositions are [Die10, Proposition 2.1] and [Die10, Proposition 2.3].

Now let $K|k$ be a field extension of degree n with basis $b_1, \ldots, b_n$, let E be an elliptic curve over K (rather than over k!), and let $\varphi : E \longrightarrow \mathbb{P}^1_K$ be a covering of degree 2 with $\varphi \circ [-1] = \varphi$.

Now let $P \in E(K)$. Let $S_{\varphi,n+1}(\mathrm{X}_1, \mathrm{Y}_1, \ldots, \mathrm{X}_n, \mathrm{Y}_n, \varphi(P))$ be a polynomial obtained by inserting the coordinates of $\varphi(P)$ for the variables X_{n+1}, Y_{n+1} in an $(n+1)^{\text{th}}$ summation polynomial of E with respect to φ; note that this polynomial is unique up to multiplication with a non-trivial constant.

Let $S^{(1)}, \ldots, S^{(n)} \in k[\mathrm{X}_1, \mathrm{Y}_1, \ldots, \mathrm{X}_n, \mathrm{Y}_n]$ be defined by

$$\sum_{j=1}^{n} b_j S^{(j)} = S_{\varphi,n+1}(\mathrm{X}_1, \mathrm{Y}_1, \ldots, \mathrm{X}_n, \mathrm{Y}_n, \varphi(P)). \tag{19}$$

Clearly, if $S^{(j)}$ is non-zero, just as $S_{\varphi,n+1}$, it is multigraded of multidegree $(2^{n-1}, \ldots, 2^{n-1})$. Note also that a different basis of $K|k$ would give rise to a system of polynomials over k which generate the same k-vector space. The same holds if the summation polynomial is multiplied by a non-trivial constant or if the coordinates of $\varphi(P)$ are simultaneously multiplied by a non-trivial constant. In particular, the subscheme $V(S^{(1)}, \ldots, S^{(n)})$ of $(\mathbb{P}^1_k)^n$ does not depend on these choices.

For $Q_1, \ldots, Q_n \in \mathbb{P}^1(k)$, the following conditions are equivalent:

- There exist $P_1, \ldots, P_n \in E(\overline{K})$ such that $P_1 + \cdots + P_n = P$ and $x(P_i) = Q_i$ for all $i = 1, \ldots, n$.
- $S_{\varphi,n+1}(Q_1, \ldots, Q_n, \varphi(P)) = 0$.
- For all $j = 1, \ldots, n$, $S^{(j)}(Q_1, \ldots, Q_n) = 0$, that is, $(Q_1, \ldots, Q_n)$ is a k-rational point of $V(S^{(1)}, \ldots, S^{(n)})$.

Definition 5.6. A tuple $(P_1, \ldots, P_n) \in E(K)^n$ with $P_1 + \cdots + P_n = P$ and $\varphi(P_i) \in \mathbb{P}^1(k)$ for $i = 1, \ldots, n$ is called a *decomposition* of P with respect to φ. Let such a decomposition be given and let $Q_i := \varphi(P_i)$. Now, if $(Q_1, \ldots, Q_n)$ is an isolated point of $V(S^{(1)}, \ldots, S^{(n)})$, the decomposition is called *φ-isolated.*

The "decomposition problem" is now the following computational problem: Given a prime power q, $n \in \mathbb{N}$, an $\mathbb{F}_q$-basis $b_1, \ldots, b_n$ of $\mathbb{F}_{q^n}|\mathbb{F}_q$, an elliptic curve E over $\mathbb{F}_{q^n}$ (given by a Weierstraß model), $\varphi : E \longrightarrow \mathbb{P}^1_k$ as well as $P \in E(\mathbb{F}_{q^n})$ of degree 2 with $[-1] \circ \varphi = \varphi$, output a list of decompositions of P with respect to φ containing all φ-isolated decompositions. A "decomposition algorithm" is then a randomized algorithm for this problem. By [Die10, Proposition 2.6] we have:

Proposition 5.7. *There exists a decomposition algorithm which operates in an expected time of $\mathcal{P}oly(e^{n^2} \cdot \log(q))$. For fixed n the expected running time is therefore polynomially bounded in $\log(q)$.*

We need a lower bound on the probability that a a uniformly randomly distributed point has a φ-isolated decomposition. For this, we impose the following condition on the covering $\varphi : E \longrightarrow \mathbb{P}^1_{\mathbb{F}_{q^n}}$ with $\deg(\varphi) = 2$ and $\varphi \circ [-1] = \varphi$ is:

Condition 5.8. There exists a point $P \in \mathbb{P}^1(\overline{\mathbb{F}}_q)$ which is a ramification point of φ such that the points $P, \sigma(P), \ldots, \sigma^{n-1}(P)$ are all distinct and φ is not ramified at $\sigma(P), \ldots, \sigma^{n-1}(P)$.

The key result for the analysis of the algorithm is now:

Proposition 5.9. *Let n be fixed. Then again for elliptic curves $E/\mathbb{F}_{q^n}$ and coverings φ such that Condition* 5.8 *holds the following is true:*

a) $\#\{P \in E(\mathbb{F}_{q^n}) \mid \varphi(P) \in \mathbb{P}^1(\mathbb{F}_q)\} \sim q$.

b) *There exist constants $C, D > 0$ such that the following holds: Let M be any subset of $\{(P_1, \ldots, P_n) \in E(\mathbb{F}_{q^n})^n \mid \varphi(P_i) \in \mathbb{P}^1(\mathbb{F}_q)$ for all $i = 1, \ldots, n\}$. Then the number of elements $P \in E(\mathbb{F}_{q^n})$ such that there exists a φ-isolated decomposition $(P_1, \ldots, P_n)$ of $\pm P$ with $P_1, \ldots, P_n \in M$ is*

$$\geq D \cdot \#M - C \cdot q^{n-1}.$$

Part a) of this result follows from [Die10, Proposition 4.10]. Part b) is a summary of [Die10, Proposition 4.28] for fixed n.

5.3. **The index calculus algorithm.** Let $n \geq 2$ be fixed. We now describe the index calculus algorithm leading to Proposition 5.1.

As already mentioned the algorithm has the same structure as the algorithm in subsection 3.2. So we describe the necessary modifications.

Let $E/\mathbb{F}_{q^n}, A, B, C_1, \ldots, C_u \in E(\mathbb{F}_{q^n})$ with $B \in \langle A \rangle$ be given.

Step 1 – Computation of the group order and factorization. We can now use Schoof's algorithm ([Sch85]) to compute $N = \#E(\mathbb{F}_{q^n})$. Of course, the running time of Step 1 is still subexponential in $\log(q)$.

Step 2 – Construction of the factor base. We choose $\varphi : E \longrightarrow \mathbb{P}^1_{\mathbb{F}_{q^n}}$ satisfying Condition 5.8. By [Die10, Proposition 2.8] except if $(q, n) \neq (3, 2)$ such a covering exists, and, moreover, it can be constructed in an expected time which is polynomially bounded in $\log(q)$. We enumerate the set $\{P \in E(\mathbb{F}_{q^n}) \mid \varphi(P) \in \mathbb{P}^1(\mathbb{F}_q)\}$.

Then we fix some subset $\mathcal{F}$ of cardinality $\lceil q^{1-\frac{1}{n}} \rceil$ or $\lceil q^{1-\frac{1}{n}} \rceil + 1$ of $\{P \in E(\mathbb{F}_{q^n}) \mid \varphi(P) \in \mathbb{P}^1(\mathbb{F}_q)\}$ which is invariant under application of $[-1]$. (Note that for q large enough this is possible by Proposition 5.9.) The set of large primes is then $\mathcal{L} := \{E(\mathbb{F}_{q^n}) \mid \varphi(P) \in \mathbb{P}^1(\mathbb{F}_q), P \notin \mathcal{F}\}$.

Clearly, these operations can be performed in an expected time of $\tilde{O}(q)$.

Step 3 – Generation of the tree of large prime relations. The construction tree of large prime relations is performed as in subsection 3.2, only that now we use the decomposition algorithm. A minor difference is that we no longer have unique factorization, and we check for each output of the decomposition algorithm if it leads to a useful FP or PP relation. The procedure is therefore as follows:

Procedure: Construction of the tree of large prime relations.

```
Construct a labeled rooted tree $T$ with vertex set contained in $\mathcal{L} \dot\cup \{*\}$ as follows:
Let $T$ consist only of the root $*$, labeled with 0.
Let $N_{\max} \longleftarrow \lceil q^{1-1/n+1/n^2} \rceil$.
Let $s \longleftarrow 1$.
Repeat
    Repeat
        Choose $s_1, \ldots, s_u \in \mathbb{Z}/N\mathbb{Z}$ uniformly and independently at random.
        Apply a decomposition algorithm to $\sum_j s_j C_j$; let $L$ be the output of
        the algorithm.
        Check for every tuple $(P_1, \ldots, P_n) \in L$ if it leads to a relation of the form
            $$\sum_j r_j F_j + Q = \sum_j s_j C_j$$
        where $Q \in \mathcal{L} - (\mathcal{F} \cup T)$   ("useful FP relation")
        or
            $$\sum_j r_j F_j + r_P P + Q = \sum_j s_j C_j$$
        where $r_P > 0, P \in T_{s-1}$ and $Q \in \mathcal{L} - (\mathcal{F} \cup T)$   ("useful PP relation").
        If this is the case,
            fix such a relation.
            If we have a "useful FP relation",
                insert $Q$ and an edge from $*$ to $Q$ into $T$
            if we have a "useful PP relation",
                insert $Q$ and an edge from $P$ to $Q$ into $T$.
            In both cases label $Q$ with $s$ and the edge with $(r_j)_j$ (in sparse
            representation).
    Until $T$ contains $2^{s-1} \cdot \lceil q^{1-1/n} \rceil$ edges or the number of edges equals $N_{\max}$.
    If the number of edges equals $N_{\max}$, STOP.
    Let $s \longleftarrow s + 1$.
```

Here as in subsection 3.2 T_s is the subtree of T consisting of vertices with label $\leq s$, that is, the tree which is constructed until including stage s.

With part b) of Proposition 5.9 the analysis in subsection 3.3 carries also easily over to the present setting. Let us first consider Stage 1 of the algorithm, that is, $s = 1$.

We set

$$M := \mathcal{F}^{n-1} \times (\mathcal{L} - T).$$

For q large enough (independently of T) this set has cardinality $\geq (q^{1-\frac{1}{n}})^{n-1} \cdot \frac{q}{4}$. Therefore the number of elements in $E(\mathbb{F}_{q^n})$ for which we obtain a new edge is $\geq \frac{D}{4} \cdot (q^{1-\frac{1}{n}})^{n-1} \cdot q - C \cdot q^{n-1} = \frac{D}{4} \cdot q^{n-1+\frac{1}{n}} - Cq^{n-1}$. For q large enough this is

$$\geq \frac{D}{8} \cdot q^{n-1+\frac{1}{n}},$$

and again for q large enough, the probability that one try (one iteration of the inner Repeat-loop) leads to a new edge is therefore

$$\geq \frac{D}{16} \cdot q^{-(1-\frac{1}{n})}.$$

Thus for q large enough, the expected number of tries until a tree of size $\lceil q^{1-\frac{1}{n}} \rceil$ is constructed is

$$\leq \frac{D}{16} \cdot (q+1)^{2-\frac{2}{n}}.$$

Let us now assume that we are in Stage s with $s \geq 2$. We set

$$M := \mathcal{F}^{n-2} \times (\mathcal{F} \cup T_{s-1}) \times (\mathcal{L} - T).$$

Now for q large enough (and independently of T, in particular, independently of s) this set has cardinality $\geq (q^{1-\frac{1}{n}})^{n-2} \cdot 2^{s-2} \cdot q^{1-\frac{1}{n}} \cdot \frac{q}{4} = 2^{s-2} \cdot (q^{1-\frac{1}{n}})^{n-1} \cdot \frac{q}{4}$. For q large enough the probability to obtain a new edge is

$$\geq \frac{D}{16} \cdot 2^{s-2} \cdot q^{-(1-\frac{1}{n})}.$$

This implies that for q large enough (independently of s) the following holds: Given any tree T with $2^{s-2} \cdot \lceil q^{1-\frac{1}{g}} \rceil$ edges, the expected number of tries until a tree T with $\min\{2^{s-1} \cdot \lceil q^{1-\frac{1}{n}} \rceil, N_{\max}\}$ edges is constructed is

$$\leq \frac{32}{D} \cdot (q+1)^{2-\frac{2}{n}}.$$

This completes the analysis.

Step 4 – relation generation. The relation generation is also as in subsection 3.2, again with the obvious difference that we use the decomposition algorithm. Again by item b) of Proposition 5.9 it is obvious that the expected running time is in $\tilde{O}(q^{2-\frac{2}{n}})$.
Steps 5–7 are as in the original algorithm.

The overall running time and conclusion. Altogether we obtain an expected running time of $\tilde{O}(q^{2-\frac{2}{n}})$, and we have storage requirements of $\tilde{O}(q^{1-\frac{1}{n}+\frac{1}{n^2}})$, as indicated in Proposition 5.1.

Acknowledgments

The author thanks Wulf-Dieter Geyer and Florian Heß for discussions on representations of divisors and Ralf Gerkmann and Alan Lauder for discussions on point counting algorithms. The author also thanks Peter Bruin for pointing out a mistake in the algorithm for the computation of uniformly randomly distributed effective divisor of a specific degree.

References

[AHU74] A. Aho, J. Hopcroft, and J. Ullman. *The design and analysis of computer algorithms.* Addison-Wesley, 1974. MR0413592 (54:1706)

[Coh96] H. Cohen. *A course in computational algebraic number theory.* Springer-Verlag, 1996. MR1228206 (94i:11105)

[Die08] C. Diem. *On arithmetic and the discrete logarithm problem in class groups of curves*, 2008. Habilitation thesis.

[Die10] C. Diem. On the discrete logarithm problem in elliptic curves. Accepted for publication at *Compos. Math.*, 2010.

[EG02] A. Enge and P. Gaudry. A general framework for subexponential discrete logarithm algorithms. *Acta. Arith.*, 102:83–103, 2002. MR1884958 (2002k:11225)

[Gau09] P. Gaudry. Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. *J. Symbolic Computation*, 44:1690–1702, 2009. MR2553574

[GTTD07] P. Gaudry, E. Thomé, N. Thériault, and C. Diem. A double large prime variation for small genus hyperelliptic index calculus. *Math. Comp.*, 76:475–492, 2007. MR2261032 (2007j:11174)

[Heß01] F. Heß. Computing Riemann-Roch spaces in algebraic function fields and related topics. *J. Symb. Comput.*, 11, 2001. MR1890579 (2003j:14032)

[Heß05] F. Heß. Computing relations in divisor class groups of algebraic curves over finite fields. preprint, ca. 2005.

[LMD90] G. Lachaud and M. Martin-Deschamps. Nombre de points des jacobiennes sur un corps fini. *Acta. Arith.*, 56:329–340, 1990. MR1096346 (92d:11065)

[LW08] A. Lauder and D. Wan. Counting points on varieties over finite fields of small characteristic. In J. Buhler and P. Stevenhagen, editors, *Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography.* Cambridge University Press, 2008. MR2467558 (2009j:14029)

[Nag07] K. Nagao. Index calculus attack for Jacobian of hyperelliptic curves of small genus using two large primes. *Japan J. Indust. Appl. Math.*, 24, 2007. MR2374992 (2008m:11243)

[Pil90] J. Pila. Frobenius maps of abelian varieties and fining roots of unity in finite fields. *Math. Comp.*, 55:745–763, 1990. MR1035941 (91a:11071)

[Pil91] J. Pila. Counting points on curves over families in polynomial time. Available on the arXiv under math.NT/0504570, 1991.

[Pom87] C. Pomerance. Fast, rigorous factorization and discrete logarithm algorithms. In D. Johnson, T. Nishizeki, A. Nozaki, and H. Wolf, editors, *Discrete Algorithms and Complexity, Proceedings of the Japan US Joint Seminar, June 4-6, 1986, Kyoto, Japan*, pages 119–143, 1987. MR910929 (88m:11109)

[RS62] J. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois J. Math.*, 6(64-94), 1962. MR0137689 (25:1139)

[Sch85] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod *p*. *Math. Comp.*, 44:483–494, 1985. MR777280 (86e:11122)

[Sem04] I. Semaev. Summation polynomials and the discrete logarithm problem on elliptic curves. available under http://eprint.iacr.org/2004/031, 2004.

[Sti93] H. Stichtenoth. *Algebraic Function Fields and Codes.* Springer-Verlag, 1993. MR1251961 (94k:14016)

University of Leipzig, Institute for Mathematics, Johannisgasse 26, 04103 Leipzig, Germany

E-mail address: diem@math.uni-leipzig.de