Implementing the Arithmetic of $C_{3,4}$ Curves

Abdolali Basiri[1,3], Andreas Enge[2], Jean-Charles Faugère[1], and Nicolas Gürel[2]

[1] Laboratoire d'Informatique de Paris 6 (CNRS/UMR 7606)
4 place Jussieu, 75252 Paris Cedex 05, France
{basiri,jcf}@calfor.lip6.fr

[2] INRIA Futurs & Laboratoire d'Informatique (CNRS/FRE 2653)
École polytechnique, 91128 Palaiseau Cedex, France*
{enge,gurel}@lix.polytechnique.fr

[3] Department of Mathematics and Computer Sciences
Damghan University of Sciences, Damghan, Iran
basiriab2@yahoo.com

Abstract. We provide explicit formulae for realising the group law in Jacobians of superelliptic curves of genus 3 and $C_{3,4}$ curves. It is shown that two distinct elements in the Jacobian of a $C_{3,4}$ curve can be added with 150 multiplications and 2 inversions in the field of definition of the curve, while an element can be doubled with 174 multiplications and 2 inversions. In superelliptic curves, 10 multiplications are saved.

1 Introduction

The interest in the arithmetic of low genus algebraic curves has been spurred by the fact that their Jacobians provide attractive groups to implement discrete logarithm based cryptosystems. While the attention first focused on the simplest types of curves, namely elliptic and hyperelliptic ones, it is shifting towards more complicated ones in the form of superelliptic and more generally $C_{a,b}$ curves. Attacks on high [5, 6] and medium genus hyperelliptic cryptosystems [9, 20] make it seem advisable to concentrate on curves of genus at most 3, which leaves superelliptic cubics and $C_{3,4}$ curves. There are a number of generic algorithms for computing $\mathcal{L}$-spaces of arbitrary curves, thus implementing the arithmetic of their Jacobian groups, like [13, 15], to cite only the most recent ones. For superelliptic and $C_{a,b}$ curves, faster special purpose algorithms have been developed, relying either on Gröbner basis computation [1] or LLL on polynomials [8, 12, 3]. None of these articles provide a precise count of the number of operations for the arithmetic of non-hyperelliptic curves of low genus.

In [2], the present authors identify special Jacobian elements, called "typical", that, while admitting a special simplified representation by two polynomials (cf. Section 2), yet cover the major part of the Jacobian. (In fact, in the cryptographic context of a large finite base field and randomly chosen elements, one

* Projet TANC, Pôle Commun de Recherche en Informatique du Plateau de Saclay, CNRS, École polytechnique, INRIA, Université Paris-Sud

does not expect to ever encounter non-typical elements.) Then two algorithms for adding typical elements are developed. The first algorithm is inspired by Cantor's algorithm for hyperelliptic curves. The second one is obtained by applying the FGLM algorithm for changing the ordering of a Gröbner basis, and yields explicit formulae for the group law in terms of operations with polynomials. These two algorithms for superelliptic cubics generalise readily to $C_{3,4}$ curves. A first quick implementation in the superelliptic case required about 250 multiplications in the underlying field for the algorithm à la Cantor, and about 200 multiplications for the formulae on the polynomial level, thus improving by a factor of 3 on our implementation of the algorithm of [8].

Flon and Oyono use a slightly different approach in [7] to obtain explicit formulae for the group law on typical elements.[1] The formulae they give for superelliptic cubics require 156 resp. 174 field multiplications (depending on whether two distinct elements are added or an element is doubled) and 2 inversions in the field. For $C_{3,4}$ curves, they announce 177 resp. 198 multiplications and again 2 inversions, without providing further details.

In the present article we show how a carefully optimised implementation of our formulae allows to save up to 16 multiplications for superelliptic curves and 27 multiplications for $C_{3,4}$ curves compared to [7]. We explain in detail how we obtained a straight line program, suited for a straightforward implementation in constrained environments such as smartcards, from the formulae manipulating polynomials, and for the first time we provide details for $C_{3,4}$ curves.

In the following section, we give a concise overview of superelliptic cubics and $C_{3,4}$ curves and relate without proof the algorithm developed in [2]. After a few remarks on the underlying field arithmetic in Section 3, we present in Section 4 algorithms for computing with low degree polynomials. These algorithms serve as a toolbox for transforming the formulae on the polynomial level into formulae on the coefficient level, which we describe together with an exact operation count in Section 5.

2 Jacobians of $C_{3,4}$ curves

Let K be a perfect field. A *superelliptic curve of genus* 3 or *Picard curve* over K is a smooth affine curve C of the form $C : Y^3 = f(X)$ with $f \in K[X]$ monic of degree 4; a $C_{3,4}$ *curve* is more general and may additionally have a term of the form $h(X)Y$ with $h \in K[X]$ of degree at most 2. The place at infinity of the function field extension $K(C)/K(X)$ corresponding to such a curve is totally ramified and rational over K, whence the K-rational part of the Jacobian of C is isomorphic as a group to the ideal class group of the coordinate ring $K[C] = K[X,Y]/(Y^3 + hY - f)$. By the Riemann–Roch theorem, each ideal class contains a unique ideal $\mathfrak{a}$ of minimal degree $\#(K[C]/\mathfrak{a})$ not exceeding 3, and this ideal is called the *reduced* representative of the class. In the following,

[1] Technically speaking, they compute a minimum with respect to the $C_{3,4}$ order in the ideal itself and end up in the inverse class, while we compute a minimum in the inverse class and end up with a reduced representative of the ideal itself.

we shall only consider ideals not containing two distinct prime ideals above a rational prime of $K[X]$. Such an ideal can be written as

$$\mathfrak{a} = (u, Y - v) \text{ with } u, v \in K[X],\ u \text{ monic}, \deg v < \deg u \text{ and } u | v^3 + hv - f;$$

its degree is exactly $\deg u$.

Most ideal classes (a proportion of $1 - \frac{1}{q}O(1)$ in the case of a finite ground field $\mathbb{F}_q$) are represented by such an ideal satisfying furthermore $\deg u = 3$ and $\deg v = 2$; we call these ideals "typical". Conversely, it is shown in [2], that for superelliptic curves of genus 3 a typical ideal is automatically reduced, and the proof carries over to $C_{3,4}$ curves. In the remainder of this article, we shall only consider typical ideals and the cases where ideals and polynomials behave in a typical way (for instance, the remainder of a polynomial of degree at least $n + 1$ upon division by a polynomial of degree n is supposed to be $n - 1$). Whenever these assumptions do not hold (which one does not expect to happen in practice), one may have recourse to a slower generic algorithm. Alternatively, one may develop specific formulae, that actually turn out to be simpler than in the common case.

The product of two ideal classes, represented by ideals $\mathfrak{a}_i = (u_i, Y - v_i)$, $\deg u_i = 3$, $\deg v_i = 2$, $i \in \{1, 2\}$, is obtained in two steps. The *composition* step corresponds to ideal multiplication and yields $\mathfrak{a} = (u, Y - v) = \mathfrak{a}_1\mathfrak{a}_2$. Here, $u = u_1u_2$ is monic of degree 6, and v of degree 5 is computed by interpolation. In the *addition* case, where $u_1 \neq u_2$ (and typically u_1 and u_2 are coprime), v is obtained by Chinese remaindering as follows:

$$s_1 = u_1^{-1} \bmod u_2;\ t = s_1(v_2 - v_1) \bmod u_2;\ v = v_1 + tu_1. \tag{1}$$

In the *doubling* case $\mathfrak{a}_1 = \mathfrak{a}_2$, a Hensel lift yields

$$\begin{aligned} &s_3 = (3v_1^2 + h)^{-1} \bmod u_1;\ w_1 = \frac{v_1^3 + hv_1 - f}{u_1};\ t = -s_3w_1 \bmod u_1; \\ &v = v_1 + tu_1. \end{aligned} \tag{2}$$

The *reduction* step takes as input an ideal $\mathfrak{a} = (u, Y - v)$ with u of degree 6 and v of degree 5, and outputs an equivalent ideal $\mathfrak{a}' = (u', Y - v')$ of degree 3, which by [2] is the reduced representative of its class. Let e be the minimum with respect to the $C_{3,4}$ order of the ideal $(u, Y^2 + vY + v^2 + h)$ in the class of $\mathfrak{a}^{-1}$, that is, e is the element whose pole at infinity has minimal multiplicity. It is shown in [2] for superelliptic curves and easily generalised to $C_{3,4}$ curves that

$$e = tY^2 + \varphi Y + \psi,$$

where the polynomial t, of degree 2, is obtained by executing two steps of the extended Euclidian algorithm on u and v, and $\varphi = tv \bmod u$; otherwise said, there is a linear polynomial s such that $su + tv = \varphi$ of degree 3. Moreover, $\psi = t(v^2 + h) \bmod u$. Then the reduced ideal $\mathfrak{a}' = \frac{e}{u}\mathfrak{a} = (u', Y - v')$ is computed as follows:

$$
\begin{aligned}
\lambda &:= \frac{t^2 f - \varphi\psi}{u} \\
\mu &:= \frac{\varphi^2 - t\psi + t^2 h}{u} \\
u' &= \frac{f\left(t(t^2 f - 3\varphi\psi) + \varphi^3\right) + \psi^3 \ + \ h\left(t\psi(th - 2\psi) + \varphi(t^2 f + \varphi\psi)\right)}{u^2} \qquad (3) \\
v' &= \mu^{-1}\lambda \bmod u'
\end{aligned}
$$

Here, all divisions by u are exact, that is with remainder zero.

3 Field arithmetic

In the previous section, we have shown how to realise the arithmetic of $C_{3,4}$ curves using a representation by polynomials. The main goal of this article is to adopt a lower level point of view and to provide formulae for the coefficients of the output polynomials in terms of those of the input polynomials. Thus, the operations we consider as elementary are operations in the field defining the curve. Our main motivation being potential cryptographic applications, we have (not too small) finite fields in mind, but the final formulae will hold for any field. However, for certain optimisations to work, we exclude fields of characteristic 2, 3 and 5. (As a side note, a superelliptic cubic is singular in characteristic 3, while in characteristic 2, it has a special structure, which might make it less attractive for cryptography, cf. [10].) There is only one division by 5 in our formulae; if need be, it could be removed at the expense of a few extra multiplications.

There are a thousand and one ways of organising the computations, so an optimality criterion is needed. Naturally, this should be the running time of a group operation in the Jacobian, which will ultimately depend on the concrete implementation and the concrete environment. A reasonable and theoretically tractable approximation is the number of elementary field operations. In many situations (over not too small finite fields, for instance, but not over the rational numbers) additions and subtractions take a negligible time compared to multiplications and inversions (divisions being realised as an inversion followed by a multiplication). Notice that multiplications by small natural numbers can be realised by a few additions, so they come for free.

Moreover, we do not count divisions by small constants (precisely, 2, 3 and 5) either. For instance, in a prime finite field $\mathbb{F}_p$, represented by $\{0, \ldots, p-1\}$, division of an element a by 2 is trivial: either a is even, then it may be divided by 2 as an integer; or it is odd, then $a + p$ is even and $\frac{a+p}{2}$ is a representative of the result in $\{0, \ldots, p-1\}$. Slightly less straightforward, a division of a by 3 may be realised by first computing the remainder of a upon division by 3. Since $4 \equiv 1 \pmod 3$, this is a matter of splitting a in blocks of two bits using bit masks and shifts and adding these base 4 digits, much as the test for divisibility by 9 of a number in decimal notation. Then, one adds the appropriate multiple of p and divides by 3. For the remainder modulo 5, an alternating sum of base 4 digits may be used. In a general finite field, represented as a vector space over $\mathbb{F}_p$, divisions by small constants can be carried out coordinate wise.

Finally, this leaves us with two variables to minimise, the number of multiplications and the number of inversions, and there is a trade off between these two. Depending on the library for finite field arithmetic, an inversion usually takes between three and ten times as long as a multiplication. We therefore tried to eliminate as many inversions as possible, as long as this introduced only a few extra multiplications. For instance, two independent inversions of field elements a and b may be replaced by one inversion and three multiplications as follows:

$$u = (a \cdot b)^{-1},\ a^{-1} = u \cdot b,\ b^{-1} = u \cdot a.$$

4 Polynomial arithmetic

Once the algorithm of Section 2 exhibited, the remaining task is no more connected to geometry, but rather to the topic of symbolic computation. The rational formulae given there are expressed in terms of operations with polynomials; it remains to phrase them in terms of their coefficients. A straightforward implementation of the polynomial arithmetic involved is of course trivial; the problem of minimising the number of field operations, however, appears to be hard. The only feasible approach we have found consists of performing local optimisations on pieces of the formulae.

In this section, we review different approaches and algorithms of polynomial arithmetic useful for this task. All of them are well-known in the symbolic computation community; however, textbooks often focus on the asymptotic behaviour of the algorithms and do not treat the very small instances we are interested in. While commenting on our choices for the concrete case of $C_{3,4}$ curves, we hope that the following overview will be helpful in further situations.

4.1 Multiplication

Let $M(m,n)$ denote the number of field multiplications carried out for multiplying two polynomials with m resp. n coefficients, that is, of degree $m-1$ and $n-1$; and let $M(n) := M(n,n)$. If useful, we indicate the employed algorithm by a subscript. For instance, the "naïve" method has $M(m,n) = mn$. A trivial improvement arises when one or both polynomials are monic; in the latter case, the equation
$(X^{m-1} + f(X))(X^{n-1} + g(X)) = X^{m+n-2} + f(X)X^{n-1} + g(X)X^{m-1} + (fg)(X)$
shows that at the expense of a few additions, only $M(m-1, n-1)$ multiplications are needed.

A substantial improvement is obtained by Karatsuba's multiplication [16]. Using the relation $(aX+b)(cX+d) = a \cdot cX^2 + \big((a+b)\cdot(c+d) - ac - bd\big)X + b \cdot d$, it achieves $M_{\mathrm{K}}(2) = 3$, and, by recursively splitting the polynomials in half,

$$M_{\mathrm{K}}(2^m, 2^n) = 2^{m-n}3^n \text{ for } m \geq n.$$

Generalisations by Toom [21] and Cook [4] to the product of two polynomials with three coefficients yield $M_{\mathrm{TC}}(3) = 5$, and the analogous recursive strategy may be applied.

4.2 Exact division

The simplest way of computing the quotient and remainder of $v = v_n X^n + v_{n-1} X^{n-1} + \cdots$ divided by $u = u_m X^m + u_{m-1} X^{m-1} + \cdots$ is the schoolbook method: invert u_m; the first term of the quotient, $u_m^{-1} v_n X^{n-m}$, is then computed with one multiplication; multiply it back by u, subtract from v; and continue in the same way.

If the remainder is of no interest or known to be zero, then it is not necessary to multiply back by all of u. In fact, it suffices to consider only the leading terms of u and v, while the lower ones determine the remainder. The k leading coefficients of the quotient are obtained with one inversion and

$$1 + \cdots + k \tag{4}$$

multiplications. If moreover u is monic, then the inversion and the multiplications by u_m^{-1} are saved, resulting in

$$1 + \cdots + (k - 1) \tag{5}$$

multiplications. Letting $k = n - m + 1$ yields the full quotient. As observed by Jebelean [14] in the case of integer division, if the division is exact, that is, the remainder is zero, then it is also possible to work with only the trailing coefficients of u and v from right to left; in fact, his algorithm amounts to the division of the reciprocal polynomials of u and v. The number of operations is unaffected, but the algorithm deploys its benefits when used to work from both sides simultaneously as suggested by Schönhage in [19], using the k lowest and $n - m + 1 - k$ highest coefficients for some k. For given u and v, the value $k = \left\lfloor \frac{n-m+1}{2} \right\rfloor$ is optimal. If the effort for computing u and v is to be taken into account, other choices may be preferable; for instance, we shall use $k = 1$ most of the time.

4.3 Short product

As seen in Section 4.2 on exact divisions, one is sometimes interested in only the trailing (or leading) coefficients of a polynomial. If this polynomial is the result of a multiplication, then these coefficients are obtained by what is known as a "short product". In the case of trailing coefficients, it can be seen as the product of truncated power series, returning upon input of two polynomials u and v of degree $n - 1$ their product modulo X^n. Instead of computing the full product of u and v and then truncating, Mulders suggested the following algorithm [17]: choose a cutoff point $k \geq \frac{n}{2}$, and write

$$u = u_0 + u_1 X^k \text{ and } v = v_0 + v_1 X^k.$$

Then $uv \bmod X^n$ is computed recursively as

$$u_0v_0 + \big((u_0v_1 \bmod X^{n-k}) + (u_1v_0 \bmod X^{n-k})\big)X^k$$

by a full and two short products. If $S(n)$ denotes the number of field multiplications required for computing a short product modulo X^n, we obtain the recursive formula

$$S(n) = M(k) + 2S(n-k).$$

Hanrot and Zimmermann [11] showed that if the full product is computed by Karatsuba's algorithm, then the optimal cutoff point k is the largest power of 2 not exceeding n. For instance, a short product modulo X^3 is computed by a full Karatsuba product of order 2 and two further field multiplications, resulting in altogether $S(3) = 5$ multiplications. This is the same number as for the full product by the Toom–Cook approach, but fewer additions and no division by 3 are required. It is to be expected that with Toom–Cook as the basic multiplication method, the optimal cutoff point will be once or twice a power of 3; then $S(4)$ becomes $S(4) = M_{\mathrm{TC}}(3) + 2S(1) = 7$. To generalise to products of polynomials of different degrees, let for $d \geq m \geq n$ the value $S(m, n; d)$ denote the number of multiplications required to compute the product modulo X^d of a polynomial of degree $m-1$ with one of degree $n-1$. Then for a cutoff point $k \leq n$ we obtain

$$S(m, n; d) = M(k) + S(m-k, k; d-k) + S(n-k, k; d-k).$$

For instance, $S(4, 3; 4)$ with Toom–Cook and $k = 3$ becomes

$$S(4, 3; 4) = M_{\mathrm{TC}}(3) + S(1, 3; 1) + S(0, 3; 1) = 5 + 1 + 0 = 6.$$

4.4 Interpolation

Karatsuba and Toom–Cook multiplication essentially work by evaluating the factors at small arguments (say, 0, 1, -1, 2, $\cdots$), multiplying the values and interpolating the result. Additionally, they treat the values at "∞", that is the leading coefficients, separately. Of course, this approach can be extended to higher degree polynomials as long as the base field has a sufficiently large characteristic so that the interpolation points are different. Evaluating a (low degree) polynomial in small integers requires only additions and multiplication by (small) constants, interpolation uses also divisions by (small) integers. So in our model, these steps cost nothing. One thus obtains a complexity of

$$M(m, n) = m + n - 1.$$

But the interpolation approach is not limited to simple multiplications; it can be extended to arbitrary polynomial and even rational formulae as long as the result is a polynomial, that is, all divisions are exact. For instance, the polynomial u' as computed in (3) is monic of degree 3, so it can be reconstructed by computing the values of $u' - X^3$ in 0, 1 and -1. Each value requires as many *field multiplications* as there are *polynomial multiplications* in the numerator of the formula, after adding parentheses and suitably reusing common subexpressions, and additionally a squaring and an inversion of the corresponding values of u in the denominator and a multiplication by the inverses. As mentioned in Section 3, the different inversions may be pooled into only one if one is willing to spend more time with multiplications.

4.5 Extended Euclidian algorithm

The classical extended Euclidian algorithm, upon input of two polynomials r_{-1} and r_0 with $\deg r_{-1} \geq \deg r_0$, computes the greatest common divisor d of r_{-1} and r_0 together with multipliers a and b such that

$$d = ar_{-1} + br_0.$$

It proceeds by iterated divisions with remainder, until the remainder vanishes, and thus requires a certain number of inversions. The greatest common divisor is only defined up to multiplication by constants, and it is possible to modify the algorithm to use pseudodivisions without field inversions (this is essentially the subresultant algorithm). Keeping track of the multipliers u and v then requires extra multiplications. Assume by induction that remainders r_{i-1} and r_i and multipliers a_{i-1}, a_i, b_{i-1} and b_i are given such that

$$r_{i-1} = a_{i-1}r_{-1} + b_{i-1}r_0 \text{ and } r_i = a_i r_{i-1} + b_i r_0;$$

the initial values being $a_{-1} = b_0 = 1$, $a_0 = b_{-1} = 0$.

Let ℓ be the function that to a polynomial associates its leading coefficient. Then the pseudodivision of r_{i-1} by r_i yields a quotient q_{i+1} and a remainder r_{i+1} such that

$$\ell(r_i)^{\deg r_{i-1} - \deg r_i + 1} r_{i-1} = q_{i+1} r_i + r_{i+1},$$

$a_{i+1} = \ell(r_i)^{\deg r_{i-1} - \deg r_i + 1} a_{i-1} - q_{i+1} a_i$ and $b_{i+1} = \ell(r_i)^{\deg r_{i-1} - \deg r_i + 1} b_{i-1} - q_{i+1} b_i$ satisfy $r_{i+1} = a_{i+1} r_{-1} + b_{i+1} r_0$.

We analyse the most common case in more detail, where $\deg r_0 = \deg r_{-1} - 1$ and the remainder degrees drop by 1 in each step, that is, all the q_i have degree 1. Letting $r_{i-1} = \alpha X^k + \beta X^{k-1} + \cdots$ and $r_i = \gamma X^{k-1} + \delta X^{k-2} + \cdots$, the next quotient is computed as $q_{i+1} = \gamma \cdot \alpha\, X + (\gamma \cdot \beta - \delta \cdot \alpha)$. This requires 3 multiplications in the general case, 1 multiplication if r_{i-1} or r_i are monic and comes for free if both of them are monic.

The next remainder is obtained as $r_{i+1} = \ell(r_i)^2 \cdot r_{i-1} - q_{i+1} \cdot r_i$, using interpolation with $1 + 2\deg r_i$ multiplications (or just $\deg r_i$ if r_i is monic).

Now, $\ell(r_i)^2$ is known, and computing a_{i+1} is free for $i = 0$, and requires 1 multiplication for $i = 1$ and $M(1, \deg a_{i-1}+1)+M(2, \deg a_i+1) = i-1+M(2, i)$ multiplications for $i \geq 2$. If furthermore r_0 is monic, then $a_2 = -q_1$ comes also for free, and $a_3 = \ell(r_2)^2 + q_1 \cdot q_2$ requires only $M(2) = 3$ multiplications.

Obtaining b_{i+1} is free for $i = 0$ and requires $M(2) = 3$ multiplications for $i = 1$ and $M(1, \deg b_{i-1} + 1) + M(2, \deg b_i + 1) = i + M(2, i + 1)$ multiplications for $i \geq 2$.

The following table summarises the number of multiplications carried out to compute the greatest common divisor, a and b, depending on the degree n of r_0, that is, the number of division steps, and on the monicity of r_{-1} and r_0. It is assumed that polynomials are multiplied using interpolation as explained in Section 4.4.

n	generic			r_{-1} monic			r_0 monic		
	gcd	a	b	gcd	a	b	gcd	a	b
1	6	0	0	4	0	0	2	0	0
2	14	1	3	12	1	3	7	0	3
3	24	5	9	22	5	9	16	3	9
4	36	11	17	34	11	17	27	9	17

4.6 Modular division by linear algebra

One ingredient in our formulae for superelliptic arithmetic is the computation of $v' = \mu^{-1}\lambda \pmod{u}$, where μ is of degree 1, λ of degree 2, and u monic of degree 3. This computation can be carried out by first determining $\mu^{-1} \bmod u$ by the extended Euclidian algorithm as described in Section 4.5 and division by the greatest common divisor, a constant in the base field; then multiplying by λ and finally reducing modulo u. In our implementation, these steps require 22 multiplications and one inversion. In this section, we describe a different approach, saving 2 multiplications. The problem to be solved is much less generic than those of the previous sections; the proposed solution, relying on linear algebra, is quite general, however, and may also be applied to different constellations of degrees.

Write $\mu = \mu_1 X + \mu_0$, $\lambda = \lambda_2 X^2 + \lambda_1 X + \lambda_0$ and $v' = x_2 X^2 + x_1 X + x_0$ with unknown x_2, x_1, x_0. Then, by degree considerations, there is a further unknown value γ such that $\mu v + \gamma u = \lambda$. Comparing coefficients, we obtain a system of four linear equations in four variables, in which the equation $\gamma = -\mu_1 x_2$ can be substituted immediately. Performing Gaussian elimination yields the solution

$$\begin{aligned} x_2 &= \alpha^{-1}\beta \\ x_0 &= \mu_0^{-1}(\lambda_0 + \mu_1 u_0 x_2) \\ x_1 &= \mu_0^{-1}(\lambda_1 + \mu_1(u_1 x_2 - x_0)) \end{aligned}$$

with

$$\begin{aligned} \alpha &= (\mu_1^2 u_1 + \mu_0^2 - (\mu_0\mu_1)u_2)\mu_0 - (\mu_1 u_0)\mu_1^2 \\ \beta &= -\lambda_1(\mu_0\mu_1) + \lambda_0\mu_1^2 + \lambda_2\mu_0^2. \end{aligned}$$

α and β are computed with 11 multiplications. Then α and μ_0 are inverted simultaneously with 3 multiplications and one inversion as described in Section 3. The computation of x_2, x_0 and x_1 then requires $1 + 2 + 3 = 6$ multiplications (reusing the expression $\mu_1 u_0$ needed for α), for a total of 20 multiplications and one inversion.

5 Explicit formulae

In this section, we develop explicit formulae for the composition and reduction steps, counting precisely the number of field multiplications and inversions. To keep the presentation readable, we use the building blocks of polynomial arithmetic of Section 4; expanding these to formulae involving only field operations is a trivial task.

We use two little tricks to speed up the computations. First, it is possible to complete the fourth power in f; after the linear change of variables $X \mapsto X - \frac{f_3}{4}$ we may assume that $f = X^4 + f_2 X^2 + f_1 X + f_0$. The impact of this observation is not very high, since f is hardly used in the formulae (it is mainly implicitly present in the relations $v^3 + vh \equiv f \pmod{u}$).

Second, the composition step involves the extended Euclidian algorithm, and the resulting greatest common divisor is normalised to be 1. This normalisation step requires an inversion, which can be saved by modifying the output of the composition to be (u, v, d) with d in the base field such that the real ideal product is given by $(u, Y - d^{-1}v)$. The composition step being of little interest *per se*, there is no harm done as long as this modification is taken into account in the reduction process.

5.1 Composition — addition

Theorem 1. *On input of two ideals $\mathfrak{a}_1 = \langle u_1, Y - v_1 \rangle$ and $\mathfrak{a}_2 = \langle u_2, Y - v_2 \rangle$ such that $u_i | v_i^3 + v_i h - f$ and $\gcd(u_1, u_2) = 1$, and assuming the typical behaviour of the remainder degrees during the Euclidian algorithm, polynomials u and v and a field element d such that $\mathfrak{a}_1\mathfrak{a}_2 = \langle u, Y - d^{-1}v \rangle$ can be computed with 37 multiplications.*

Proof. We first compute $u = u_1 u_2$ by Toom–Cook multiplication with 5 field multiplications. Then, implementing (1), we determine s_1 of degree 2 and d such that $s_1 u_1 \equiv d \pmod{u_2}$ by applying the extended Euclidian algorithm to u_2 and $u_1 - u_2$. According to the table in Section 4.5, with $n = 2$ and r_{-1} monic this needs $12 + 3 = 15$ multiplications. We then compute $t_1 = s_1(v_2 - v_1)$ with 5 multiplications and the quotient q of the result by u_2 with 1 multiplication according to (5). The polynomial $t = t_1 - q \cdot u_2$ of degree 2 is then obtained by interpolation on three points with 3 multiplications for the values of $q \cdot u_2$. Finally, $v = d \cdot v_1 + u_1 \cdot t$ is computed with $M(1,3) + M(3) = 8$ multiplications.

5.2 Composition — doubling

With the preparations of Section 4, the composition part of doubling is as straightforward as addition, but it requires noticeably more operations.

Theorem 2. *On input of an ideal $\mathfrak{a}_1 = \langle u_1, Y - v_1 \rangle$ such that $u_1 | v_1^3 + v_1 h - f$, and assuming the typical behaviour of the remainder degrees during the Euclidian algorithm, polynomials u and v and a field element d such that $\mathfrak{a}_1^2 = \langle u, Y - d^{-1}v \rangle$ can be computed with 61 multiplications.*

Proof. We use the notation introduced in (2). First, we compute $u = u_1^2$ and v_1^2 with 5 multiplications each. To obtain w_1, we start with the four highest coefficients of $v_1^3 + v_1 h - f = v_1^2 \cdot (v_1 + h) - f$ using $S(4, 3; 4) = 6$ multiplications (see Section 4.3). An exact division by u_1 yields w_1 with 6 multiplications as shown in (5). The next step is to determine s_3 of degree 2 and d in the base field such that $s_3 \cdot (3v_1^2 + h) \equiv d \pmod{u_1}$. By the table in Section 4.5, this requires 19 multiplication since $n = 3$ and r_{-1} is monic.

We then reduce w_1, which is of degree 3, modulo the monic u_1 by subtracting the appropriate multiple of u_1, obtained with 3 multiplications; multiply by s_3 with $M(3) = 5$ multiplications and reduce again modulo u_1, which takes 1 multiplication for the quotient and 3 multiplications for the remainder, yielding t in a total of 12 multiplications.

Finally, v is obtained multiplying v_1 by d and t by u with $M(3, 1) + M(3) = 8$ multiplications.

Notice that for adding as well as for doubling, the composition step is not more costly on $C_{3,4}$ than on superelliptic curves.

5.3 Reduction

Theorem 3. *On input of an ideal* $\mathfrak{a} = \langle u, Y - d^{-1}\tilde{v} \rangle$ *such that* $u | (d^{-1}\tilde{v})^3 + d^{-1}\tilde{v}h - f$, u *monic of degree* 6, $\tilde{v}$ *of degree* 5, *the reduced representative* $\mathfrak{a}' = \langle u', Y - v' \rangle$ *in the ideal class of* $\mathfrak{a}$ *can be computed with* 113 *multiplications and* 2 *inversions. In the superelliptic case,* 10 *multiplications may be saved.*

Proof. To facilitate keeping track of the total number of multiplications, from time to time we provide their balance, having the number for the superelliptic case precede that for the general $C_{3,4}$ case.

$$\boxed{0/0}$$

We use the notation of Section 2. Let furthermore $v = d^{-1}\tilde{v}$, and denote the coefficient of a polynomial in front of X^i by a subscript i. The first step of the algorithm consists of finding the minimum $e = tY^2 + \varphi Y + \psi$ of $\mathfrak{a}$ with respect to the $C_{3,4}$ order, where the polynomial t, of degree 2, is obtained by executing two steps of the extended Euclidian algorithm on u and v, $\varphi = tv \bmod u$ is of degree 3 and $\psi = tv^2 + th \bmod u$ is of degree 5. Notice that e is defined only up to multiplication by constants; we shall compute the representative with leading coefficient 1 for the $C_{3,4}$ order, that is, with ψ monic. Inspection of (3) shows that then also u' will be monic, and no further normalisation will be needed.

To avoid inverting d, we shall use $\tilde{v}$ in the place of v, and correct the polynomials later on. Thus, we determine $\tilde{t}$ of degree 2, $\tilde{\varphi}$ of degree 3 and a linear polynomial ζ such that $\tilde{\varphi} = \tilde{t}\tilde{v} \bmod u = t \cdot \tilde{v} - \zeta \cdot u$. Using the algorithm of Section 4.5, the computation of $\tilde{t}$ and ζ requires 17 multiplications. By carrying out the Euclidian algorithm symbolically and simplifying the resulting formulae by hand, one may save 2 squarings as follows. The δ_i designate temporary variables.

$$
\begin{aligned}
\delta_1 &= u_4 \cdot \tilde{v}_5 - \tilde{v}_3\\
\delta_2 &= u_5 \cdot \tilde{v}_5 - \tilde{v}_4\\
\delta_3 &= \delta_1 \cdot \tilde{v}_5 - \delta_2 \cdot \tilde{v}_4\\
\tilde{t}_2 &= \delta_3 \cdot \tilde{v}_5\\
\zeta_1 &= t_2 \cdot \tilde{v}_5\\
\delta_4 &= \delta_3 \cdot u_5\\
\delta_5 &= \delta_2 \cdot \tilde{v}_3 + \delta_4 - \tilde{v}_5 \cdot (\tilde{v}_5 \cdot u_3 - \tilde{v}_2)\\
\tilde{t}_1 &= \delta_5 \cdot \tilde{v}_5\\
\zeta_0 &= \tilde{v}_5 \cdot (t_1 - \delta_2 \cdot \delta_3)\\
\tilde{t}_0 &= (\delta_5 - \delta_4) \cdot \delta_2 + \delta_1 \cdot \delta_3
\end{aligned}
$$

$\boxed{15/15}$

The polynomial $\tilde{\varphi}$ is obtained via interpolation from $\tilde{t}$, $\tilde{v}$, u and ζ with 8 multiplications. Then, we compute polynomials $\tilde{\psi}$ and ξ such that

$$\tilde{\psi} = \tilde{\varphi}\tilde{v} \bmod u = \tilde{\varphi} \cdot \tilde{v} - \xi \cdot u,$$

the correction by the additional term th in the definition of ψ being postponed. Computing the 3 leading coefficients of $\tilde{\varphi}\tilde{v}$ takes $S(3) = 5$ multiplications, and the three coefficients of the quotient ξ by the monic u are obtained with 3 multiplications. Then $\tilde{\psi}$ may be computed by interpolation on six points with 12 multiplications.

$\boxed{43/43}$

Since we worked with $\tilde{v}$ instead of v, the polynomials $\tilde{t}$, $\tilde{\varphi}$ and $\tilde{\psi}$ have to be adjusted by powers of d, at the same time as making $\tilde{\psi}$ monic. We profit from the inversion of $\tilde{\psi}_5$ by computing simultaneously the inverse of u_0, which will be needed later on, with 3 multiplications and one field inversion as described in Section 3. Then, we obtain the minimum e via

$$t = (\psi_5^{-1} d \cdot d) \cdot \tilde{t},\ \varphi = (\psi_5^{-1} \cdot d) \cdot \tilde{\varphi},\ \psi = \psi_5^{-1} \cdot \tilde{\psi} + th.$$

As will become clear in the following, we do in fact not need the coefficients φ_1, ψ_1 and ψ_2, whence this step can be carried out with 11 multiplications in the superelliptic case. When $h \neq 0$, the computation of $t_0 h_0$ requires an additional multiplication, and ψ_3 and ψ_4 need the two leading terms of th, obtained with $S(2) = 3$ multiplications.

$\boxed{57/61}$

Define the polynomials λ and μ of degree 2 and 1, respectively, as in the equations before (3). In the following, we shall perform polynomial arithmetic "from both sides" as described in Section 4.2. All constant coefficients are computed separately. For instance, λ_0 and μ_0 are obtained with 7 multiplications if $h = 0$, including those by the already computed u_0^{-1}. For $C_{3,4}$ curves, μ_0 requires additionally the computation of $t_0^2 h_0$, which is obtained with one extra multiplication since $t_0 h_0$ has already been used for ψ_0.

For $\mu_1 = -t_2$, there is nothing to do. Taking into account that $f_4 = 1$, $f_3 = 0$ and $\psi_5 = 1$, the numerator of λ starts with

$$(t_2^2 - \varphi_3)X^8 + (2t_1 \cdot t_2 - \varphi_2 - \varphi_3 \cdot \psi_4)X^7 + \cdots,$$

and these coefficients are computed with 3 multiplications. The two leading terms of the quotient by u require another multiplication, so that the total number of multiplications for λ and μ becomes 11 in the superelliptic and 12 in the $C_{3,4}$ case.

$\boxed{68/73}$

The polynomial u' of the result is computed via (3). For $h = 0$, the constant coefficient u'_0 is easily seen to be computable with 7 multiplications, reusing values like φ_0^2 already needed for λ_0 or μ_0. If $h \neq 0$, then the term

$$h_0 \cdot \big(t_0\psi_0 \cdot (t_0 h_0 - 2\psi_0) + \varphi_0 \cdot (t_0^2 f_0 + \varphi_0\psi_0)\big)$$

requires only the 3 additional multiplications marked with a dot.

$\boxed{75/83}$

For the high degree part of u', we compute the leading terms of the numerator $X^{15} + \alpha X^{14} + \beta X^{13} + \cdots$ of (3) as

$$\begin{aligned} \alpha &= t_2 \cdot (\lambda_2 - 2(\varphi_3 + h_2)) + 3\psi_4, \\ \beta &= 3(t_1 \cdot \lambda_2 + \psi_3 + \psi_4^2) - t_2 \cdot (3\varphi_2 + 2h_1) + (\varphi_3^2 - 3t_2 \cdot \psi_4) \cdot \varphi_3 \\ &\quad + h_2 \cdot \big(t_2^2 \cdot (h_2 + \varphi_3) + (\varphi_3^2 - 3t_2\psi_4) - t_2\psi_4 - 2t_1\big), \end{aligned}$$

where we have used the relation $\lambda_2 = t_2^2 - \varphi_3$. These quantities are obtained with 7 multiplications in the superelliptic case and 9 multiplications in the case of $C_{3,4}$ curves. Then the leading coefficients of u' are given by

$$u'_3 = 1,\ u'_2 = \alpha - 2u_5,\ u'_1 = \beta - u_5 \cdot (2u'_2 + u_5) - 2u_4$$

with 1 multiplication.

$\boxed{83/93}$

Finally, v' is computed as $v' = \mu^{-1}\lambda \bmod u$ with 20 multiplications and one inversion as described in Section 4.6.

$\boxed{103/113}$

In the following table, we summarise the number of multiplications carried out by our algorithm for adding or doubling divisors, totalling the efforts for the composition and the reduction step. We distinguish between ordinary multiplications and squarings in the base field. While we did not pursue this distinction in the present article due to space restrictions, separating these two numbers is a simple exercise. The number of inversions is always 2.

	superelliptic			$C_{3,4}$		
	mult.	sqr.	m.+s.	mult.	sqr.	m.+s.
addition	129	11	140	139	11	150
doubling	143	21	164	153	21	174

6 Concluding remarks

Formulae for the arithmetic of hyperelliptic curves of genus 3 are reported in [18]. They require 76 field multiplications and one inversion for adding two distinct elements, and 71 multiplications and one inversion for doubling an element. While our formulae for superelliptic and $C_{3,4}$ curves need more operations, the factor of only about 2 shows that $C_{3,4}$ curves constitute a reasonable alternative to hyperelliptic curves for cryptographic use.

Availability of the formulae

The MAGMA code of our formulae can be downloaded from the web at the address

`http://www.lix.polytechnique.fr/Labo/Andreas.Enge/C34.html`

Acknowledgement. Thanks to Pierrick Gaudry for his comments on our work.

References

1. Seigo Arita. Algorithms for computations in Jacobian group of C_{ab} curve and their application to discrete-log based public key cryptosystems. *IEICE Transactions*, J82-A(8):1291–1299, 1999. In Japanese. English translation in the proceedings of the Conference on The Mathematics of Public Key Cryptography, Toronto 1999.
2. Abdolali Basiri, Andreas Enge, Jean-Charles Faugère, and Nicolas Gürel. The arithmetic of Jacobian groups of superelliptic cubics. Rapport de Recherche 4618, INRIA, November 2002. Available at http://www.inria.fr/rrrt/rr-4618.html, to appear in *Mathematics of Computation*.
3. Mark L. Bauer. The arithmetic of certain cubic function fields. *Mathematics of Computation*, 73(245):387–413, 2004.
4. Stephen A. Cook. *On the Minimum Computation Time of Functions*. PhD thesis, Harvard University, 1966.
5. Andreas Enge. Computing discrete logarithms in high-genus hyperelliptic Jacobians in provably subexponential time. *Mathematics of Computation*, 71(238):729–742, 2002.

6. Andreas Enge and Pierrick Gaudry. A general framework for subexponential discrete logarithm algorithms. *Acta Arithmetica*, 102(1):83–103, 2002.
7. Stéphane Flon and Roger Oyono. Fast arithmetic on Jacobians of Picard curves. Preprint, Cryptology ePrint Archive 2003/079, available at http://eprint.iacr.org/2003/079, 2003.
8. S. D. Galbraith, S. M. Paulus, and N. P. Smart. Arithmetic on superelliptic curves. *Mathematics of Computation*, 71(237):393–405, 2002.
9. Pierrick Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 19–34, Berlin, 2000. Springer-Verlag.
10. Pierrick Gaudry and Nicolas Gürel. An extension of Kedlaya's point counting algorithm to superelliptic curves. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 480–494, Berlin, 2001. Springer-Verlag.
11. G. Hanrot and P. Zimmermann. A long note on Mulders' short product. *Journal of Symbolic Computation*, 37(3):391–401, 2004.
12. Ryuichi Harasawa and Joe Suzuki. Fast Jacobian group arithmetic on C_{ab} curves. In Wieb Bosma, editor, *Algorithmic Number Theory — ANTS-IV*, volume 1838 of *Lecture Notes in Computer Science*, pages 359–376, Berlin, 2000. Springer-Verlag.
13. Floran Heß. Computing Riemann–Roch spaces in algebraic function fields and related topics. *Journal of Symbolic Computation*, 33(4):425–445, 2002.
14. Tudor Jebelean. An algorithm for exact division. *Journal of Symbolic Computation*, 15:169–180, 1993.
15. Kamal Khuri-Makdisi. Linear algebra algorithms for divisors on an algebraic curve. *Mathematics of Computation*, 73(245):333–357, 2004.
16. А. Карацуба and Ю. Офман. Умножение многозначных чисел на автоматах. *Доклады Академии Наук СССР*, 145(2):293–294, 1962. English translation: Multiplication of Multidigit Numbers on Automata, *Soviet Physics — Doklady*, 7(7):595–596, 1963.
17. Thom Mulders. On short multiplications and divisions. *Applicable Algebra in Engineering, Communication and Computing*, 11:69–88, 2000.
18. Jan Pelzl, Thomas Wollinger, Jorge Guajardo, and Christof Paar. Hyperelliptic curve crypstosystems: Closing the performance gap to elliptic curves. In Colin D. Walter, Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 351–365, Berlin, 2003. Springer-Verlag.
19. Arnold Schönhage and Ekkehart Vetter. A new approach to resultant computations and other algorithms with exact division. In Jan van Leeuwen, editor, *Algorithms — ESA'94*, volume 855 of *Lecture Notes in Computer Science*, pages 448–459, Berlin, 1994. Springer-Verlag.
20. Nicolas Thériault. Index calculus attack for hyperelliptic curves of small genus. In Chi Sung Laih, editor, *Advances in Cryptology — ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 75–92, Berlin, 2003. Springer-Verlag.
21. А. Л. Тоом. О сложности схемы из функциональных элементов, реализирующей умножение целых чисел. *Доклады Академии Наук СССР*, 150(3):496–498, 1963. English translation: The complexity of a scheme of functional elements realizing the multiplication of integers, *Soviet Mathematics*, 4:714–716, 1963.