Efficient Hyperelliptic Arithmetic using Balanced Representation for Divisors

Steven D. Galbraith[1], Michael Harrison[2], and David J. Mireles Morales[1]

[1]Mathematics Department
Royal Holloway, University of London
{steven.galbraith, d.mireles-morales}@rhul.ac.uk

[2]School of Mathematics and Statistics
University of Sydney
mch@maths.usyd.edu.au

Abstract. We discuss arithmetic in the Jacobian of a hyperelliptic curve C of genus g. The traditional approach is to fix a point $P_\infty \in C$ and represent divisor classes in the form $E - d(P_\infty)$ where E is effective and $0 \leq d \leq g$. We propose a different representation which is balanced at infinity. The resulting arithmetic is more efficient than previous approaches when there are 2 points at infinity.
This is the full version of the paper.

1 Introduction

The study of efficient addition algorithms for divisors on genus 2 curves has come to a point where cryptography based on these curves provides an alternative to its well-established elliptic curve counterpart. The most commonly used case is when the curve has 1 point at infinity and addition corresponds to Cantor's ideal composition and reduction algorithm in [3]. Explicit formulae have been given by Lange in [13] and a comprehensive account of the different addition algorithms can be found in [4].

It is then only natural to extend this work to hyperelliptic curves with 2 points at infinity since curves with a rational Weierstrass point are rare among all hyperelliptic curves. Further motivation is given by pairing based cryptography, since Galbraith, Pujolas, Ritzenthaler and Smith gave in [8] an explicit construction of a pairing-friendly genus 2 curve C which typically cannot be given a model with 1 point at infinity. It is an interesting question to determine how efficiently pairings can be implemented for these curves.

Scheidler, Stein and Williams [16] gave algorithms to compute in the so-called infrastructure of a function field (also see [10]). Their approach included composition and reduction algorithms used by Cantor, as well as an algorithm that had no analogue in his theory, known as a "baby-step". The relationship between the infrastructure and divisor class groups was studied by Paulus and Rück [14]. It is well-known that arithmetic on curves with two points at infinity is slower than the simpler case of one point at infinity (our methods do not change this).

In this article we view the Cantor and infrastructure algorithms as operations on the Mumford representation of affine effective semi-reduced divisors, rather than as operations on the Jacobian of a curve. This simple change of perspective suggests a representation of elements in the Jacobian of C which is more "balanced" at infinity. We therefore show that arithmetic in the Jacobian may be performed more efficiently than done by [5, 9, 14, 15]. In the case of genus 2 curves, all explicit addition formulae presented so far [5] can be used with our representation, giving improved results (see Table 1).

We interpret the algorithms developed for the infrastructure, in particular the baby-step, from our new perspective. This gives, in our opinion, a simpler explanation of them. In particular, we do not need to discuss continued fraction expansions. Note however that we only discuss the application of these ideas to arithmetic in the Jacobian, rather than computation in the infrastructure itself. We observe that computing inverses of elements using an unbalanced representation is non-trivial, whereas with our representation it is easy. Previous literature (e.g., [5]) has suggested that the baby step has no analogue for curves with one point at infinity; however we explain that one can develop a fast baby step operation in all settings.

We would like to point out that the group law for hyperelliptic curves with 2 rational points at infinity for the computer algebra system Magma [2], implemented by the second author, follows the approach described in this article. It was first released in Magma V2.12, in July 2005.

2 Divisor class groups of hyperelliptic curves

In this paper we consider a genus g hyperelliptic curve C defined over a field K given by a non-singular planar model

$$y^2 + h(x)y = F(x) = \sum_{i=0}^{2g+2} F_i x^i,$$

where $h(x), F(x) \in K[x]$ satisfy $\deg(F) \leq 2g+2$ and $\deg(h) \leq g+1$. If $P = (x, y)$ is a point on C, the point $(x, -h(x) - y)$ also lies on C, we will call this point the *hyperelliptic conjugate* of P and we will denote it by $\overline{P}$.

If $F_{2g+2} = 0$ and $\deg(h) \leq g$, then C will have one K-rational point at infinity, in this case we say that this is an *imaginary* model for C. If $F_{2g+2} \neq 0$ then C will have two points at infinity, possibly defined over a quadratic extension of K, in this case we say that C is represented by a *real* model. If the curve C has a K-rational point we can always move it to the line at infinity so that the points at infinity of the curve are K-rational.

Let C be an algebraic curve defined over a field K. All divisors considered in this article will be K-rational unless otherwise stated. Denote by $\mathrm{Div}^0(C)$

the group of degree zero K-rational divisors on C. Two divisors D_0 and D_1 are *linearly equivalent*, denoted $D_0 \equiv D_1$, if there is a function f such that

$$\operatorname{div}(f) = D_1 - D_0,$$

where $\operatorname{div}(f)$ is the divisor of f.

Definition 1. *The* divisor class group *of C is the group of K-rational divisor classes modulo linear equivalence. We will denote it as* $\mathrm{Cl}(C)$*. The class of a divisor D in* $\mathrm{Cl}(C)$ *will be denoted by $[D]$. We define* $\mathrm{Cl}^0(C)$ *as the degree zero subgroup of* $\mathrm{Cl}(C)$*.*

Definition 2. *We say that an effective divisor $D = \sum_i P_i$ is* semi-reduced *if $i \neq j$ implies $P_i \neq \overline{P}_j$. We say that a divisor D on a curve of genus g is* reduced *if it is semi-reduced, and has degree $d \leq g$. Throughout this article we will denote the degree of a divisor D_i as d_i.*

There is a standard way to represent an effective affine semi-reduced divisor D_0 on a hyperelliptic curve C: Mumford's representation. In this case we will represent our divisor using a pair of polynomials $u(x), v(x) \in K[x]$, where $u(x)$ is a polynomial of degree d_0 whose roots are the X-coordinates of the points in D_0 (with the appropriate multiplicity) and u divides $F - hv - v^2$. This last condition implies that if x_i is a root of u, the linear polynomial $v(x_i)$ gives the Y-coordinate of the corresponding point in D_0. Because of this last condition, D_0 must be a semi-reduced divisor. We will denote the divisor associated to the pair of polynomials $u(x)$ and $v(x)$ as $\operatorname{div}[u, v]$. Notice that Mumford's representation can be used to describe any effective affine semi-reduced divisor. Describing elements of $\mathrm{Cl}^0(C)$ is a more delicate matter.

To describe elements of $\mathrm{Cl}^0(C)$ we will need a degree g effective divisor D_∞. Throughout this article, unless otherwise stated, this divisor will be as below.

Definition 3.
- *If C has a unique point at infinity ∞, then $D_\infty = g\infty$.*
- *If g is even and C has two points at infinity ∞^+ and ∞^- then $D_\infty = \frac{g}{2}(\infty^+ + \infty^-)$.*
- *If g is odd and C has two points at infinity, then $D_\infty = \frac{g+1}{2}\infty^+ + \frac{g-1}{2}\infty^-$. In this case we will further assume that ∞^+ and ∞^- are K-rational points.*

Proposition 1. *Let D_∞ be a K-rational degree g divisor, and let $D \in \mathrm{Div}^0(C)$ be a K-rational divisor on the curve C. Then $[D]$ has a unique representative in* $\mathrm{Cl}^0(C)$ *of the form $[D_0 - D_\infty]$, where D_0 is an effective K-rational divisor of degree g whose affine part is reduced.*

Proof. The case $D_\infty = g\infty^+$ is Proposition 4.1 of [14]. Now let D_∞ be any degree g divisor. If D is a representative of a class in $\mathrm{Cl}^0(C)$, using Proposition 4.1 in [14] we know that $D + (D_\infty - g\infty^+) \equiv D_1 - g\infty^+$, where D_1 is an effective degree g divisor with affine reduced part. This implies that $D \equiv D_1 - D_\infty$ and proves existence.

To prove uniqueness, suppose that D_1 and D_2 are two effective degree g divisors with affine reduced support, and $D_1 - D_\infty \equiv D_2 - D_\infty$. Adding $D_\infty - g\infty^+$ to both sides gives $D_1 - g\infty^+ \equiv D_2 - g\infty^+$. Proposition 4.1 from [14] implies that $D_1 = D_2$. □

A small problem from a computational point of view is that this proposition does not guarantee that the supports of D_0 and D_∞ are disjoint, and indeed, in some cases they will have points in common which should be "cancelled out". However, divisors of the form $D_0 - D_\infty$ with D_0 and D_∞ having disjoint support are generic, so it is enough to describe their arithmetic for many applications. In this article we will give a complete addition algorithm for hyperelliptic curves, that becomes very efficient in the generic case.

If the curve C has two different points at infinity ∞^+ and ∞^-, it is possible to prove that the function y/x^{g+1} is well defined and not zero at each of ∞^+ and ∞^-. One can further prove that

$$\frac{y}{x^{g+1}}(\infty^+) \neq \frac{y}{x^{g+1}}(\infty^-),$$

so if we define

$$a_+ = (y/x^{g+1})(\infty^+), \quad a_- = (y/x^{g+1})(\infty^-),$$

it follows that $a_+ \neq a_-$. Hence, for $p(x)$ a polynomial of the form $p(x) = (a_+ x^{g+1} + \sum_{0 \le i \le g} b_i x^i)$, the function $y - p(x)$ will have valuation strictly larger than $-(g+1)$ at ∞^+ and valuation $-(g+1)$ at ∞^-.

Definition 4. *In the notation of the previous paragraph, among all degree $g+1$ polynomials with leading coefficient a_+, there is a unique polynomial in $\overline{K}[x]$ for which the valuation of the function at ∞^+ is maximal; we will denote this polynomial by H^+. Define the polynomial H^- analogously.*

If $C(x, y)$ is the equation of the curve, then $H^+(x)$ and $H^-(x)$ are the polynomials with leading coefficient a_+ and a_- such that $C(x, H^\pm(x))$ has minimal degree. Their coefficients can thus be found recursively. The polynomials $H^\pm(x)$ are just a technical tool to specify a point at infinity, similar to the choice of sign when computing the square root of a complex number. Note that the polynomials $H^\pm$ are defined over K if and only if the points ∞^+ and ∞^- are K-rational.

Definition 5. *Given two divisors D_1 and D_2, we will denote the set of pairs of integers ω^+, ω^- such that*

$$D_1 \equiv D_2 + \omega^+\infty^+ + \omega^-\infty^-,$$

as $\omega(D_1, D_2)$. We say that the numbers ω^+ and ω^- are counterweights *for D_1 and D_2 if $(\omega^+, \omega^-) \in \omega(D_1, D_2)$.*

The set $\omega(D_1, D_2)$ may be empty. If $[\infty^+ - \infty^-]$ is a torsion point on $\mathrm{Cl}^0(C)$, and the set $\omega(D_1, D_2)$ is not empty, then it is infinite; however this will not affect our algorithms. Given two divisors D_1 and D_2, calculating the values of the counterweights relating them is a difficult problem. When these values are needed in our algorithms, there will be a simple way to calculate them.

3 Operations on the Mumford Representation

In this section we recall some well-known algorithms due to Cantor [3] for computing with divisor classes of hyperelliptic curves. We will analyse them as operations on the Mumford representation of an affine semi-reduced divisor. Our main contribution is to give a geometric interpretation of these algorithms.

Algorithm 1 Composition

```
INPUT: Semi-reduced affine divisors $D_1 = \mathrm{div}[u_1, v_1]$ and $D_2 = \mathrm{div}[u_2, v_2]$.
OUTPUT: A semi-reduced affine divisor $D_3 = \mathrm{div}[u_3, v_3]$ and a pair $(\omega^+, \omega^-)$, such that
    $(\omega^+, \omega^-) \in \omega(D_1 + D_2, D_3)$.
Compute $s$ (monic), $f_1, f_2, f_3 \in K[x]$ such that
    $s = \gcd(u_1, u_2, v_1 + v_2 + h) = f_1 u_1 + f_2 u_2 + f_3(v_1 + v_2 + h).$
Set $u_3 := u_1 u_2 / s^2$ and $v_3 := (f_1 u_1 v_2 + f_2 u_2 v_1 + f_3(v_1 v_2 + F))/s \mod u_3$
**return** $\mathrm{div}[u_3, v_3]$ and $(\deg(s), \deg(s))$.
```

The result D_3 of Algorithm 1 will be denoted $D_3, (\omega^+, \omega^-) = \mathrm{comp}(D_1, D_2)$. The divisor of the function s from Algorithm 1 is

$$\mathrm{div}(s) = D_1 + D_2 - D_3 - \frac{d_1 + d_2 - d_3}{2}(\infty^+ + \infty^-), \tag{1}$$

which proves that

$$(\omega^+, \omega^-) \in \omega(D_1 + D_2, D_3).$$

Algorithm 1 is also known as *divisor composition.*

Given an affine semi-reduced divisor D_0, of degree $d_0 \geq g + 2$, Algorithm 2 finds another affine semi-reduced divisor D_1 with smaller degree d_1, and a pair of integers (ω^+, ω^-) such that

$$(\omega^+, \omega^-) \in \omega(D_0, D_1) \tag{2}$$

Algorithm 2 is known as *divisor reduction.*

The result D_1 of Algorithm 2 will be denoted as $D_1, (\omega^+, \omega^-) = \mathrm{red}(D_0)$. The geometric interpretation of Algorithm 2 is very simple: given the effective affine divisor $D_0 = \mathrm{div}[u_0, v_0]$, we know (by definition of the Mumford representation) that the divisor of zeros D_z of the function $y - v_0(x)$ has (in the notation of Algorithm 2) $D_z = D_0 + \overline{D}_1$, and if $\deg(u_0) \geq g + 2$, then the degree of D_z satisfies $\deg(D_z) < 2\deg(D_0)$, hence $\deg(D_1) < \deg(D_0)$, and if the leading term of v_0 is different to that of $H^{\pm}$ we have

$$\mathrm{div}\left(\frac{y - v_0(x)}{u_0}\right) = \overline{D}_0 - \overline{D}_1 - \frac{d_0 - d_1}{2}(\infty^+ + \infty^-). \tag{3}$$

It follows that

$$D_0 - D_1 \equiv \frac{d_0 - d_1}{2}(\infty^+ + \infty^-).$$

Algorithm 2 Reduction

```
INPUT: A semi-reduced affine divisor $D_0 = \mathrm{div}[u_0, v_0]$, with $d_0 \geq g + 2$.
OUTPUT: A semi-reduced affine divisor $D_1 = \mathrm{div}[u_1, v_1]$ and a pair $(\omega^+, \omega^-)$, such that
    $d_1 < d_0$ and Equation (2) holds.
Set $u_1 := (v_0^2 + hv_0 - F)/u_0$ made monic.
Let $v_1 := (-v_0 - h) \mod u_1$.
if the leading term of $v_0$ is $a_+ x^{g+1}$ (in the notation of Definition 4) then
    Let $(\omega^+, \omega^-) := (d_0 - g - 1, g + 1 - d_1)$.
else if the leading term of $v_0$ is $a_- x^{g+1}$ then
    Let $(\omega^+, \omega^-) := (g + 1 - d_1, d_0 - g - 1)$.
else
    Let $(\omega^+, \omega^-) := (\frac{d_0 - d_1}{2}, \frac{d_0 - d_1}{2})$.
end if
return $\mathrm{div}[u_1, v_1], (\omega^+, \omega^-)$.
```

A similar analysis when the leading coefficient of v_0 coincides with that of $H^\pm$ shows that if $D_1, (\omega^+, \omega^-) = \mathrm{red}(D_0)$, then we always have $(\omega^+, \omega^-) \in \omega(D_0, D_1)$.

If C is an imaginary model of a curve with point at infinity ∞, this relation degenerates into

$$D_0 - D_1 \equiv (d_0 - d_1)\infty. \tag{4}$$

In this case, if D_0 is a divisor of degree $d_0 = g + 1$, Algorithm 2 will produce a divisor of degree $d_1 < d_0$, satisfying Equation (4).

Algorithm 3 Composition at Infinity and Reduction

```
INPUT: A semi-reduced affine divisor $D_0 = \mathrm{div}[u_0, v_0]$ of degree $d_0 \leq g + 1$.
OUTPUT: A reduced affine divisor $D_1 = \mathrm{div}[u_1, v_1]$ and a pair of integers $(\omega^+, \omega^-)$
    such that $(\omega^+, \omega^-) \in \omega(D_0, D_1)$.
$v_1' := H^\pm + (v_0 - H^\pm \mod u_0)$,
$u_1 := (v_1'^2 + hv_1' - F)/u_0$ made monic.
$v_1 := -h - v_1' \mod u_1$.
if $H^+$ was used then
    Let $(\omega^+, \omega^-) := (d_0 - g - 1, g + 1 - d_1)$.
else if $H^-$ was used then
    Let $(\omega^+, \omega^-) := (g + 1 - d_1, d_0 - g - 1)$.
end if
return $\mathrm{div}[u_1, v_1], (\omega^+, \omega^-)$.
```

Algorithm 3 is only defined for affine semi-reduced divisors on curves given by a real model. If it were applied on a divisor of degree at least $g + 2$, Algorithm 3 would coincide with Algorithm 2. When applied on a divisor D_0 degree at most $g + 1$, Algorithm 3 can be interpreted as composing the divisor D_0 with some divisor at infinity, followed by Algorithm 2. The polynomial v_1' in this algorithm is the equivalent to polynomial v_3 in Algorithm 1. The result D_1 of this algorithm will be denoted as $D_1, (\omega^+, \omega^-) = \mathrm{red}_\infty(D_0)$. Formally, the action of this algorithm is given by the following.

Proposition 2. *Given an effective semi-reduced divisor with affine support D_0, with Mumford representation* $\mathrm{div}[u_0, v_0]$ *and degree* $d_0 \leq g+1$*. If* $D_1, (\omega^+, \omega^-) = \mathrm{red}_\infty(D_0)$*, then*

$$(\omega^+, \omega^-) \in \omega(D_0, D_1).$$

Proof. We will only prove this when the algorithm is applied using H^+. Notice that the polynomial $v_1'(x)$ has the property that the function $f = y - v_1'(x)$ has all the points in D_0 in its divisor of zeros.

The $(g+1) - d_0$ highest degree coefficients of $v_1'(x)$ coincide with those of $H^+(x)$, so the function

$$(v_1'(x))^2 + hv_1'(x) - F(x),$$

which finds the affine support of f, has degree at most $g + d_0$, and it follows that the affine support of f has at most $g + d_0$ points.

We know that the function $y - v_1'(x)$ will have valuation $-(g+1)$ at ∞^-. The divisor of f is then:

$$\mathrm{div}(f) = D_0 + D_2 - (d_0 + d_2 - (g+1))\infty^+ - (g+1)\infty^- \qquad (5)$$

If we denote by D_1 the hyperelliptic conjugate of D_2, we know that

$$\mathrm{div}(u_1) = D_2 + D_1 - d_2(\infty^+ + \infty^-)$$

which together with Equation (5) implies

$$\frac{y - v_1'(x)}{u_1} = D_0 - D_1 - (d_0 - (g+1))\infty^+ - (g+1-d_2)\infty^- \qquad (6)$$

which trivially becomes

$$D_0 \equiv D_1 + (d_0 - (g+1))\infty^+ + (g+1-d_1)\infty^-. \qquad (7)$$

The proposition follows at once. □

Remark 1. When dealing with explicit computations, the divisors D_0 and D_1 will very often have degree g, in which case we can re-write Equation (7) as

$$D_0 + (\infty^+ - \infty^-) \equiv D_1.$$

Choosing any degree g base divisor D_∞ to represent the points on the class group of C, this equation tells us that

$$(D_0 - D_\infty) + (\infty^+ - \infty^-) \equiv (D_1 - D_\infty),$$

in other words, Algorithm 3 is nothing but addition of $\infty^+ - \infty^-$; this turns out to be such a simple operation because the divisor composition is elementary and can easily be incorporated in the divisor reduction process, which is itself very simple.

We would like to emphasize that Algorithm 3 is independent of the choice of base divisor, so one has the freedom to choose a divisor D_∞ optimal in each specific case.

Remark 2. We have just seen that Algorithm 3 generically corresponds to addition of $\infty^+ - \infty^-$, however, it has long been claimed that this operation[1] has no analogue in the imaginary curve case. Using the previous remark, we propose the following.

Let $C : y^2 = G(x)$, where $\deg(G(x)) = 2g + 1$, be a non-singular imaginary model for a hyperelliptic curve of genus g. Take a point $P = (x_P, y_P)$ on C. Given an effective affine divisor $D = \mathrm{div}[u_0, v_0]$ on C, where $\deg(v_0) < \deg(u_0)$, define a P-baby step on D as follows:

$$\begin{aligned} a &= (y_P - v_0(x_P))/u_0(x_P) \\ \tilde{v}_1(x) &= au_0(x) + v_0(x) \\ u_1(x) &= \frac{(\tilde{v}_1)^2 - G(x)}{(x - x_P)u_0(x)} \\ v_1(x) &= -\tilde{v}_1 \mod u_1(x) \end{aligned}$$

The result of applying a P-baby step on the divisor D_0 is, generically, a divisor D_1 such that $D_0 + ([P] - \infty) = D_1$. This algorithm will fail when P is in the support of D_0. Doing some precomputations and using an appropriate implementation, this operation should be as efficient as a baby step. A good choice of P (for instance, having a very small x_P, or even $x_P = 0$) could have a big impact on the efficiency of this algorithm.

In the Appendix, Algorithm 6 gives explicit formulae to calculate a P-baby step in genus 2 curves, where the point P has the form $P = (0, y)$. The operation count of Algorithm 6 is (1I,1S,5M), which is very competitive compared with the (1I,2S,4M) required by its analogue in curves given by a real model [5].

The following technical lemma will be used in the next section to prove that our proposed addition algorithm finishes. It can be safely ignored by readers interested only in the computational aspects of the paper.

Lemma 1. *Let D_0 be an effective divisor of degree $d_0 = 2g$ and D_1 be an effective affine divisor of degree $d_1 \leq g$. If $(\omega_1^+, \omega_1^-) \in \omega(D_0, D_1)$,*

$$D_2, (\omega_r^+, \omega_r^-) = \mathrm{red}_\infty(D_1) \qquad \textit{(using } H^+\textit{)},$$

and we denote $(\omega_2^+, \omega_2^-) = (\omega_1^+ + \omega_r^+, \omega_1^- + \omega_r^-)$, then $(\omega_2^+, \omega_2^-) \in \omega(D_0, D_2)$ and

$$\omega_1^+ - \omega_1^- > \omega_2^+ - \omega_2^-.$$

If $\omega_1^- < (g-1)/2$ then $\omega_2^+ \leq g/2$.

[1] Some authors call it a "baby step", see Section 4.1

Proof. From the hypotheses we know that $\omega_1^+ + \omega_1^- = 2g - d_1$. Proposition 2 says that

$$(\omega_r^+, \omega_r^-) = (d_1 - (g+1), g+1-d_2), \tag{8}$$

this implies that

$$\omega_1^+ - \omega_1^- = \omega_2^+ - \omega_2^- + (2g+2-d_0-d_1),$$

which proves the first assertion. Equation (8) together with $\omega_1^+ = 2g - d_1 - \omega_1^-$ implies

$$\begin{aligned} \omega_2^+ &= \omega_1^+ + d_1 - g - 1 \\ &= (2g - d_1 - \omega_1^-) + d_1 - g - 1 \\ &= g - 1 - \omega_1^- \end{aligned}$$

by hypothesis $\omega_1^- < (g-1)/2$, so that $\omega_2^+ > (g-1)/2$, and since ω_2^+ is an integer, the result follows. □

Remark 3. Previous authors have used the notation "baby steps" and "giant steps". We explain these using our notation. Given two divisors $D_1 = \mathrm{div}[u_1, v_1]$ and $D_2 = \mathrm{div}[u_2, v_2]$ on C, a *"giant step"* on D_1 and D_2 is the result of computing $D_3 = \mathrm{comp}(D_1, D_2)$ and succesively applying reduction steps (using a red_∞ reduction) on the result until the degree of $\mathrm{red}_\infty^i(D_3)$ is at most g. "Baby steps" are only defined on reduced affine effective divisors, and the result of a "baby step" on a reduced divisor D is the divisor $\mathrm{red}_\infty(D)$.

In [9], an algorithm is given to efficiently compute a giant step. It can then be used in any arithmetic application that requires such an operation, regardless of the representation of divisors in $\mathrm{Cl}^0(C)$ being used.

4 Addition on Real Models

Throughout this section C will denote a genus g hyperelliptic curve defined over a field K, given by the equation

$$C : y^2 + h(x)y = F(x),$$

where $F(x)$ is a degree $2g+2$ polynomial. If $\mathrm{char}(K) \neq 2$, then we will further assume that $h = 0$. If $\mathrm{char}(K) = 2$, then h will be monic and $\deg(h) = g+1$.

We will also assume that the divisor D_∞ from Definition 3 is K-rational. This condition holds automatically for even g. For odd values of g one needs to further assume that the leading coefficient of F is a square in K if $\mathrm{char}(K) \neq 2$ or that the leading coefficient of F is of the form $\omega^2 + \omega$ if $\mathrm{char}(K) = 2$.

Every element $[a_0]$ of $\mathrm{Cl}^0(C)$ has a unique representative of the form $a_0 = D_0 - D_\infty$, where D_0 is a degree g effective divisor with reduced affine part. Any effective, degree g divisor D_0 can be uniquely written as $D_0 = D_0' + n_0\infty^+ + m_0\infty^-$, where D_0' is the affine support of D_0, and $n_0, m_0 \in \mathbb{Z}_{\leq 0}$; in this case we will denote the divisor $D_0 - D_\infty$ as $\mathrm{div}([u_0, v_0], n_0)$, where $\mathrm{div}[u_0, v_0] = D_0'$ is the Mumford representation of D_0'. This representation of a divisor is unique.

We would like to remark that in the notation we have just described for divisors, we always have $\deg v_0 < \deg u_0$ and n_0 is an integer such that $0 \leq n_0 \leq g - \deg(u_0)$. The implementation used in Magma represents elements of the class group as $\langle u, v', d \rangle$, where $\mathrm{div}[u, v' \mod u]$ is the Mumford representation of an affine reduced divisor and d is an even integer such that $\deg(u) \leq d \leq g+1$. We do not have enough space to describe this notation, for which we refer the reader to the Magma documentation. The element represented in Magma as $\langle u, v', d \rangle$ corresponds in our notation to the divisor $\mathrm{div}([u, v' \mod u], n)$, where n is an integer given by:

$$\begin{array}{ll} n = \lceil \frac{g-d}{2} \rceil, & \text{if } d = \deg(u) \text{ or } \deg(v' - H^- \leq g). \\ n = \lceil \frac{g-(-1)^g d}{2} \rceil - \deg(u), & \text{otherwise.} \end{array}$$

The representation used in Magma is sub-optimal for cryptographic applications since it can have $\deg(v') \geq \deg(u)$.

Given two divisors $a_1 = \mathrm{div}([u_1, v_1], n_1)$ and $a_2 = \mathrm{div}([u_2, v_2], n_2)$ of $\mathrm{Cl}^0(C)$, we want to find $a_3 = \mathrm{div}([u_3, v_3], n_3)$ such that

$$[a_1] + [a_2] = [a_3].$$

To fix notation, let

$$\begin{aligned} a_i &= \mathrm{div}[u_i, v_i] + n_i\infty^+ + m_i\infty^- - D_\infty, \\ \tilde{D}_i &= \mathrm{div}[u_i, v_i] + n_i\infty^+ + m_i\infty^-, \\ D_i &= \mathrm{div}[u_i, v_i] \end{aligned}$$

for $i \in 1, 2$.

Algorithm 4 Divisor Addition

```
INPUT: Divisors $a_i = \mathrm{div}([u_i, v_i], n_i)$ for $i \in \{1, 2\}$.
OUTPUT: $a_3 = \mathrm{div}([u_3, v_3], n_3)$, $[a_3] = [a_1] + [a_2]$.
Set $(\omega^+, \omega^-) := (n_1 + n_2, m_1 + m_2)$.
Let $D, (a, b) := \mathrm{comp}(D_1, D_2)$. Update $(\omega^+, \omega^-) := (\omega^+ + a, \omega^- + b)$.
while $\deg(D) > g + 1$ do
    $D, (a, b) := \mathrm{red}(D)$. Update $(\omega^+, \omega^-) := (\omega^+ + a, \omega^- + b)$.
end while
while $\omega^+ < g/2$ or $\omega^- < (g-1)/2$ do
    $D, (a, b) := \mathrm{red}_\infty(D)$. Update $(\omega^+, \omega^-) := (\omega^+ + a, \omega^- + b)$.
    Use $H^+$ in $\mathrm{red}_\infty$ if $\omega^+ > \omega^-$, else use $H^-$.
end while
Let $E := D + \omega^+\infty^+ + \omega^-\infty^- - D_\infty$.
Now $E$ is an effective degree $g$ divisor. Write $E = D + n_3\infty^+ + m_3\infty^-$, where $D$
    is an effective affine divisor.
return $\mathrm{div}(D, n_3)$.
```

Some comments are in order. Throughout the algorithm we always have that $(\omega^+, \omega^-) \in \omega(\tilde{D}_1 + \tilde{D}_2, D)$. We have mentioned that if $\deg(D) \geq g+2$ then $\deg(\mathrm{red}(D)) < \deg(D)$, so step 3 always finishes. Lemma 1 proves that step 4, and hence the algorithm, always finish.

Cantor's addition algorithm for curves given by an imaginary model (see [3]) can be seen as a degenerate case of our algorithm. We can think of Algorithm 4 as: 1. Divisor composition; 2. Reduction steps until the degree is at most $g+1$; 3. Use red_∞ to balance the divisor at infinity. Since imaginary models have a unique point at infinity, to perform divisor addition it suffices to compute the composition and reduction steps, making the balancing step redundant. In the following section we will argue that our divisor D_∞ is the correct choice to have an algorithm analogous to that of Cantor.

If C has even genus, the points ∞^+ and ∞^- are not K-rational and the divisors a_1 and a_2 are K-rational, by a simple rationality argument the counterweights will always be equal, hence the addition algorithm will get a divisor D with equal counterweights such that $\deg(D) \leq g$ in step 3. Algorithm 4 will then finish and step 4 will not be necesary. In this case the (non K-rational) polynomials $H^\pm$ will not be used and no red_∞ step will be computed.

This last observation suggests that, given a hyperelliptic curve C with even genus, one should move two non K-rational points to infinity and get an addition law completely analogous to Cantor's algorithm. This trivial trick could greatly simplify the arithmetic on C.

One key operation in an efficiently computable group is element inversion. Algorithm 5 describes this operation in $\mathrm{Cl}^0(C)$.

Algorithm 5 Divisor Inversion

```
INPUT: A divisor $a_1 = \mathrm{div}([u_1, v_1], n_1)$.
OUTPUT: A divisor $a_2 = \mathrm{div}([u_2, v_2], n_2)$ such that $[a_1] = -[a_2]$.
if $g$ is even then
    return $\mathrm{div}([u_1, (-h - v_1 \mod u_1)], g - \deg(u_1) - n_1)$.
else if $g$ is odd and $n_1 > 0$ then
    return $\mathrm{div}([u_1, (-h - v_1 \mod u_1)], g - m_1 - \deg(u_1) + 1)$.
else
    Let $D_1 = \mathrm{red}_\infty(\mathrm{div}[u_1, -h - v_1])$.
    return $\mathrm{div}(D_1, 0)$.
end if
```

Given the geometric analysis that we have made of the addition algorithm, computing pairings on the class group of an arbitrary hyperelliptic curve can be done following Miller's algorithm. There is not enough space in this paper to give a complete description of an algorithm to compute pairings, but Miller's functions can be calculated from Equations (1),(3) and (6).

4.1 Other proposals

Previous proposals for addition algorithms on hyperelliptic curves given by a real model use $D_\infty = g\infty^+$ instead of the divisor D_∞ we used in the previous section [14, 15]. In particular, this implies that the points ∞^+ and ∞^- need to be K-rational.

A simple modification of Algorithm 4 can be used to add divisors in $\mathrm{Cl}^0(C)$ using $D_\infty = g\infty^+$ as base divisor. All one needs to do is change the finishing condition in step 4 from $((\omega^+ < g/2)$ or $(\omega^- < (g-1)/2))$ to $(\omega^+ < g)$. Indeed, one can verify that using Algorithm 4 with a modified terminating condition coincides with the addition algorithms presented in [14, 15].

We will now compare the two proposals for addition algorithms on $\mathrm{Cl}^0(C)$. Since the performance of the algorithms, specially for cryptographic applications, will depend exclusively on its behaviour when adding generic divisors, we will restrict our analysis to this case.

Assume for a moment that the curve C has even genus g, and that D_1 and D_2 are two effective affine divisors of degree g. Generically, the result D_3 of applying succesive reductions to $\mathrm{comp}(D_1, D_2)$ until the degree is at most $g+1$ is a divisor D_3 of degree g. If this is the case, we have

$$D_1 + D_2 \equiv D_3 + (g/2)(\infty^+ + \infty^-), \tag{9}$$

Notice that the counterweights between $D_1 + D_2$ and D_3 are equal, this is a consequence of Equation (4). Using Equation (9) with $D_\infty = (g/2)(\infty^+ + \infty^-)$, we get

$$D_1 - D_\infty + D_2 - D_\infty \equiv D_3 - D_\infty,$$

which means that we have found the result of adding $D_1 - D_\infty$ and $D_2 - D_\infty$, and no "composition at infinity and reduction" steps were necessary.

If instead we work with a divisor at infinity $D'_\infty = g\infty^+$, Equation (9) becomes

$$D_1 - D'_\infty + D_2 - D'_\infty = D_3 - D'_\infty - (g/2)(\infty^+ - \infty^-),$$

so typically one will need $g/2$ extra red_∞ steps to find D_4 such that

$$D_4 - D'_\infty = (D_1 - D'_\infty) + (D_2 - D'_\infty),$$

it is not difficult to see that the need for the red_∞ steps is related to the fact that the valuations of D'_∞ at the two points at infinity are so different.

Now consider a curve C of odd genus g, and let again D_1 and D_2 be degree g affine divisors. Typically, the result after step 2 in Algorithm 4 on the divisors D_1 and D_2 will be a divisor D_3 of degree $g+1$ such that

$$D_1 + D_2 \equiv D_3 + \frac{g-1}{2}(\infty^+ + \infty^-). \tag{10}$$

Again, the counterweights between $D_1 + D_2$ and D_3 are equal as a consequence of Equation (4), and if we now compute $D_4 = \mathrm{red}_\infty(D_3)$, then generically

$$D_3 \equiv D_4 + \infty^-,$$

which together with Equation (10) gives us

$$D_1 + D_2 \equiv D_4 + \frac{g+1}{2}\infty^+ + \frac{g-1}{2}\infty^-. \tag{11}$$

Using our base divisor $D_\infty = (g+1)/2\infty^+ + (g-1)/2\infty^-$, we get

$$D_1 - D_\infty + D_2 - D_\infty \equiv D_4 - D_\infty,$$

and only one red_∞ step was needed. Notice that in this case the addition algorithm consists of composition, a series of standard reduction steps, and the last step is a single application of red_∞.

Using the base divisor $D'_\infty = g\infty^+$, Equation (10) becomes

$$D_1 - D'_\infty + D_2 - D'_\infty \equiv D_3 - D'_\infty - (g-1)/2(\infty^+ - \infty^-),$$

so one will typically need $(g-1)/2$ extra steps to find D_4 such that

$$D_4 - D'_\infty = (D_1 - D'_\infty) + (D_2 - D'_\infty).$$

Again, the need for the red_∞ steps stems from the difference in the valuations of D_∞ at both points at infinity.

Remark 4. If the curve C has odd genus g, one could try to recover a balanced representation of its elements by representing them as $D_0 - (g+1)/2(\infty^+ + \infty^-)$, where D_0 is a degree $g+1$ effective divisor with reduced affine part. This representation could be useful in the intermediate steps of a given implementation, since the use of the red_∞ algorithm would be limited to the final step of the calculation. When divisors are represented using this approach, the number of coefficients needed to describe them increases, and the composition formulae get more complicated. A detailed comparison of the performance of these methods would be interesting.

We have seen that using a “balanced” divisor at infinity, generically the number of red_∞ steps needed to compute the addition of two divisor classes in $\mathrm{Cl}^0(C)$ is 0 when g is even and 1 when g is odd; whereas when using a non-balanced divisor, the number of red_∞ steps needed to compute the addition of two divisors is generically $g/2$ for even g and $(g-1)/2$ for odd g.

In order to compare the two proposals for arithmetic in $\mathrm{Cl}^0(C)$, we must also consider the computation of inverses, a fundamental operation in a computable group which has, surprisingly, been ignored in the literature. Besides its trivial use to invert divisors, this operation is fundamental to achieve fast divisor multiplication through signed representations.

We will just analyse inversion in the generic case. To do this let D be a degree g affine effective divisor on C. Assume for a moment that g is even. The inverse of the divisor $P = D - (g/2)(\infty^+ + \infty^-)$ is the divisor $\overline{D} - (g/2)(\infty^+ + \infty^-)$, whereas if we now assume that g is odd, the divisor

$$(D - \frac{g+1}{2}\infty^+ - \frac{g-1}{2}\infty^-) + (\overline{D} - \frac{g-1}{2}\infty^+ - \frac{g+1}{2}\infty^-)$$

is principal, which means that $\overline{D} - (g-1)/2\infty^{+} - (g+1)/2\infty^{-}$ is the inverse of P, and in order to fix the divisor at infinity, using Proposition 2 it is easy to see that generically only one application of Algorithm 3 will suffice. In other words, using the "balanced" representation at infinity, 0 or 1 applications of Algorithm 3 will be needed, depending on the parity of g.

We now analyze the computation of inverses using $D'_{\infty} = g\infty^{+}$ as base divisor. Clearly, the divisor

$$(D - g\infty^{+}) + (\overline{D} - g\infty^{-})$$

is principal, so we need to find an appropriate representative of the divisor class $[\overline{D} - g\infty^{-}]$. Again, this can be done through g applications of Algorithm 3, as can be easily seen using Proposition 2.

It is now clear that computing the inverse of a divisor class is easier when the divisor at infinity is as balanced as possible, supporting our claim that a "balanced" representation is a closer analogue to that of Cantor for imaginary models, where the inverse of a divisor is its hyperelliptic conjugate, just as in our case when the genus of C is even.

Table 1 gives the cost of addition and doubling in a genus 2 curve using the explicit formulae for Algorithms 1, 2 and 3 presented in [5]. If $S = M$ and $I = 4M$ then balanced representations give a saving of around 15% for addition and 13% for doubling (if $I = 30M$ the savings become 62% and 58% respectively). The extra operations in the non-balanced case come from an additional application of Algorithm 3 in each case.

	Imaginary	Balanced	Non-balanced
Addition	1I, 2S, 22M [13]	1I, 2S, 26M	2I, 4S, 30M
Doubling	1I, 5S, 22M [13]	1I, 4S, 28M	2I, 6S, 32M
Inversion	0	0	2I, 4S, 8M

Table 1. Operation counts for genus 2 arithmetic using formulae of [5] .

5 Conclusion

We have given an explicit geometric interpretation of Algorithm 3, which made it clear that all the composition and reduction algorithms presented in this paper (all of which have been known for a long time) really act on semi-reduced affine divisors rather than on elements of $\mathrm{Cl}^0(C)$; that is to say, they can be seen as acting on the Mumford representation of a divisor. Having made this simple observation, a number of interesting consequences follow. One such observation is that in order to get simple arithmetic operations one needs to find an optimal base divisor D_{∞}, and we have argued that in cryptography-related applications the optimal choice is a balanced divisor D_{∞}. When the genus of the curve is even, if the points at infinity are non-rational (which can always be achieved), using a balanced base divisor yields an algorithm identical to that of Cantor, where the rationality takes care of the counterweights; this is impossible to achieve with non-balanced divisors.

The question of finding explicit addition formulae for curves in real representation using our proposed divisor already has an answer: since generic addition formulas have been given for Algorithms 1, 2 and 3 in a genus 2 real curve [5], we can use these formulas to calculate an addition law on $\mathrm{Cl}^0(C)$ by just changing the divisor at infinity one is working with. All the explicit addition formulae presented so far (specially for $g = 2$) that we have knowledge of (including those of [5, 9]) first compute the composition of the two affine divisors in the summands, then find the divisor with degree at most $g + 1$ which is the result of successively applying reduction steps, and finally give an explicit form of Algorithm 3. Hence, it is possible to use these formulae to compute divisor addition using our proposal with no alterations.

Appendix

In Remark 2 we mentioned that the analogue of a baby step for curves given by an imaginary model would be the addition of a point of the form $P - \infty$, where P has a special form. In Algorithm 6, we present an optimized algorithm to compute these "imaginary baby steps" in a genus 2 curve C defined over the field k by

$$C : y^2 = x^5 + \sum_{i=0}^{4} f_i x^i,$$

where we further assume that f_0 is a square in k, so that the point $P = (0, y_b)$ on C is k-rational. Divisors will be represented by their Mumford representation $(x^2 + u_1 x + u_0, v_1 x + v_0)$.

Algorithm 6 Imaginary Baby Step

```
INPUT: $D = \mathrm{div}(x^2 + u_1 x + u_0, v_1 x + v_0)$ .
OUTPUT: $D_2 = \mathrm{div}(u_2, v_2)$ such that $[D_2] = [D] + [(0, y_b) - \infty]$.
$\mu := (y_b - v_0)/u_0$ (1I,1M).
$b1 := f_4 - \mu^2 - u_1$ (1S).
$b_0 := f_3 - u_0 - 2\mu v_1 + u_1(u_1 - \mu^2 - f_4)$ (2M).
$c_1 := v_1 + \mu(u_1 - b_1)$ (1M).
$c_0 := v_0 + \mu(u_0 - b_0)$ (1M).
return $\mathrm{div}(x^2 + b_1 x + b_0, -c_1 x - c_0)$.
```

The cost of a "real baby step" in genus 2 is (1I,2S,4M) [5] and Algorithm 6 needs (1I,1S,5M), it is therefore a very efficient analogue of Algorithm 3 for curves given by an imaginary model.

We would like to point out that the idea of using degenerate divisors (a divisor is degenerate if it has less than g affine points in its support) has been considered before. Katagi et.al. [11, 12] analysed the advantages of using degenerate divisors in cryptographyc applications, and the use of degenerate divisors to optimise pairing computations has also been discussed in [1, 6, 7]. However, the use of a point P with a special form in the degenerate divisor seems to have been overlooked.

Acknowledgments

We would like to thank Mike Jacobson and the anonymous referees for their helpful comments. The first author is supported by EPSRC Grant EP/D069904/1. The third author thanks CONACyT for its financial support.

References

1. Barreto, P. S. L. M., Galbraith, S. D., Ó hÉigeartaigh, C., and Scott, M. Efficient pairing computation on supersingular abelian varieties. *Des. Codes Cryptography 42*, 3 (2007), 239–271.
2. Bosma, W., Cannon, J., and Playoust, C. The Magma algebra system. I. The user language. *J. Symbolic Comput. 24*, 3-4 (1997), 235–265. Computational algebra and number theory (London, 1993).
3. Cantor, D. G. Computing in the Jacobian of a hyperelliptic curve. *Math. Comp. 48*, 177 (1987), 95–101.
4. Cohen, H., Frey, G., Avanzi, R., Doche, C., Lange, T., Nguyen, K., and Vercauteren, F., Eds. *Handbook of elliptic and hyperelliptic curve cryptography.* Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2006.
5. Erickson, S., Jacobson, M. J., Shang, N., Shen, S., and Stein, A. Explicit formulas for real hyperelliptic curves of genus 2 in affine representation. In *WAIFI* (2007), C. Carlet and B. Sunar, Eds., vol. 4547 of *Lecture Notes in Computer Science*, Springer, pp. 202–218.
6. Frey, G., and Lange, T. Fast bilinear maps from the tate-lichtenbaum pairing on hyperelliptic curves. In *ANTS* (2006), F. Hess, S. Pauli, and M. E. Pohst, Eds., vol. 4076 of *Lecture Notes in Computer Science*, Springer, pp. 466–479.
7. Galbraith, S. D., Hess, F., and Vercauteren, F. Hyperelliptic pairings. In *Pairing* (2007), T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, Eds., vol. 4575 of *Lecture Notes in Computer Science*, Springer, pp. 108–131.
8. Galbraith, S. D., Pujolas, J., Ritzenthaler, C., and Smith, B. Distortion maps for genus two curves. *Journal of Mathematical Cryptology 3*, 1 (2009), 1–18.
9. Jacobson, M., Scheidler, R., and Stein, A. Fast arithmetic on hyperelliptic curves via continued fraction expansions. In *Advances in Coding Theory and Cryptography* (2007), T. Shaska, W. Huffman, D. Joyner, and V. Ustimenko, Eds., vol. 3 of *Series on Coding Theory and Cryptology*, World Scientific Publishing, pp. 201–244.
10. Jacobson, M. J., Scheidler, R., and Stein, A. Cryptographic protocols on real hyperelliptic curves. *Adv. Math. Commun. 1*, 2 (2007), 197–221.

11. Katagi, M., Akishita, T., Kitamura, I., and Takagi, T. Some improved algorithms for hyperelliptic curve cryptosystems using degenerate divisors. In *ICISC* (2004), C. Park and S. Chee, Eds., vol. 3506 of *Lecture Notes in Computer Science*, Springer, pp. 296–312.
12. Katagi, M., Kitamura, I., Akishita, T., and Takagi, T. Novel efficient implementations of hyperelliptic curve cryptosystems using degenerate divisors. In *WISA* (2004), C. H. Lim and M. Yung, Eds., vol. 3325 of *Lecture Notes in Computer Science*, Springer, pp. 345–359.
13. Lange, T. Formulae for arithmetic on genus 2 hyperelliptic curves. *Appl. Algebra Engrg. Comm. Comput. 15*, 5 (2005), 295–328.
14. Paulus, S., and Rück, H.-G. Real and imaginary quadratic representations of hyperelliptic function fields. *Math. Comp. 68*, 227 (1999), 1233–1241.
15. Paulus, S., and Stein, A. Comparing real and imaginary arithmetics for divisor class groups of hyperelliptic curves. In *Algorithmic number theory (Portland, OR, 1998)*, vol. 1423 of *Lecture Notes in Comput. Sci.* Springer, Berlin, 1998, pp. 576–591.
16. Scheidler, R., Stein, A., and Williams, H. C. Key exchange in real quadratic congruence function fields. *Designs, Codes and Cryptography 7* (1996), 153–174.