Efficient arithmetic on low-genus curves

Efficient Arithmetic on Low-Genus Curves

Peter Birkner

Efficient Arithmetic on Low-Genus Curves

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op maandag 16 februari 2009 om 16.00 uur

door

Peter Birkner

geboren te Hamminkeln, Duitsland

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr. T. Lange
en
prof.dr. D.J. Bernstein

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Birkner, Peter

Efficient Arithmetic on Low-Genus Curves / door Peter Birkner. –
Eindhoven: Technische Universiteit Eindhoven, 2009
Proefschrift. – ISBN 978-90-386-1517-2
NUR 919
Subject heading: Cryptology
2000 Mathematics Subject Classification: 94A60, 11G05, 11G20, 14H45, 14Q05

Printed by Printservice Technische Universiteit Eindhoven
Cover design by Verspaget & Bruinink, Nuenen
Photograph “halved apple” by Christian Hüls, idea by Relinde Jurrius

Promotor: prof.dr. T. Lange
Promotor: prof.dr. D.J. Bernstein (University of Illinois at Chicago)

Commissie:
prof.dr.dr.h.c. G. Frey (Universität Duisburg-Essen)
prof.dr. S. Galbraith (Royal Holloway, University of London)
prof.dr.ir. H.C.A. van Tilborg
prof.dr. A. Blokhuis

Contents

1. Introduction

In this thesis we investigate the arithmetic and algorithmic impact of elliptic and hyperelliptic curves of genus 2 and 3 to cryptography and to integer factorisation. When elliptic and hyperelliptic curves are used for cryptographic applications, those are almost always based on the hardness of the discrete-logarithm problem (DLP) in the group of points on an elliptic curve or in the Picard group of a hyperelliptic curve of genus 2 or 3. In general, given a group $G = \langle P \rangle$ and some $Q \in G$, the discrete-logarithm problem is to find the integer k such that $Q = [k]P$, where $[k]P$ stands for the k-fold addition of P to itself.

Cryptographers are looking for groups where this problem is secure against currently known attacks. In the present work we will show that the discrete-logarithm problem on divisor class groups of hyperelliptic curves of genus 2 and 3 over binary fields is so hard to solve, that it provides enough security to be used for real-world applications.

The most important operation in curve-based cryptosystems is *scalar multiplication* $[k]P$, where P is a point on an elliptic curve or a divisor class of a hyperelliptic curve and k an integer. To compute a scalar multiple one usually uses algorithms that involve the computation of the double of a group element or the addition of two different group elements. To increase the performance of cryptosystems based on this, much effort was put into speeding up the doubling operation and the addition operation. In the present work we suggest using divisor class halving instead of divisor class doubling. This provides a noticeable speedup in certain settings.

In the next chapter we thoroughly introduce the divisor class group of hyperelliptic curves and explain the group operation. Further background is provided in Appendix A where we explain currently known attacks.

For hyperelliptic curves of genus 2 over finite fields of characteristic 2 we provide efficient explicit halving formulas that can (for certain curves) outperform the doubling formulas and therefore replace the usually used double-and-add algorithms by halve-and-add algorithms. In Chapter 3 we investigate hyperelliptic curves of genus 2 over binary fields in detail. We classify these curves depending on their 2-rank and give for each case efficient doubling and halving formulas. The formulas are given in affine coordinates and require the computation of inverse elements in the base field. In some situations (e.g. in hardware implementations) this is not desired since inversion is always the most costly field operation, and not only in terms of time but also in terms of chip area. To provide efficient arithmetic for those implementations we give efficient doubling formulas in new

and recent coordinates which are completely inversion-free. This can be achieved at the cost of a higher number of multiplications but the overall cost is lower compared to implementations with inversions.

In Chapter 4 we have extended the halving results of the genus-2 curves to genus 3. We derive a similar classification according to the 2-rank of the hyperelliptic curves and investigate efficient halving formulas for each case. It turns out that for curves of Type III (i.e. $h(x) = x$ in the curve equation) the computation of one halving is noticeably faster than of one doubling. In the worst case one halving takes 1 field inversion, 25 field multiplications, 4 squarings and 7 square-root extractions whereas the appropriate doubling takes 1 field inversion, 44 field multiplications and 6 squarings [GKP04]. We also provide a full case study where we give halving formulas for all cases, including all special ones which occur with lower probability.

The following two chapters are dedicated to Edwards and twisted Edwards curves. We introduce these curves and investigate important properties. A very interesting result is that elliptic curves in Montgomery form are birationally equivalent to twisted Edwards curves, i.e. these curves are isomorphic to each other except for a finite number a points. With this equivalence we can bring the speed of the Edwards addition law to all Montgomery-form elliptic curves.

In Chapter 5 we also provide efficient formulas to perform the group law on Edwards curves. We look at the affine case and give also formulas in projective and inverted Edwards coordinates which are entirely inversion-free.

In the last chapter, we focus on Lenstra's elliptic curve method (ECM) for integer factorisation and on Edwards curves over the rational numbers. We show how the performance of integer factorisation can be improved by using Edwards curves. We give several methods to construct Edwards curves that are suitable for ECM. We find parametrisations to generate infinitely many of such suitable curves. First experiments have already shown a noticeable speedup compared to ECM using elliptic curves in Weierstraß form.

2. Hyperelliptic Curves

In this chapter we provide important notions for the understanding of cryptosystems based on the discrete-logarithm problem on hyperelliptic curves over finite fields. One of the most important terms of this thesis is "hyperelliptic curve". We first give an abstract definition of this term and in the following lemma a more concrete characterisation of those hyperelliptic curves that we will work with. The reader will also be familiarised with divisors and principal divisors, Picard group, Mumford representation, function fields, Cantor's algorithm and the p-rank of a hyperelliptic curve.

For the understanding of this chapter, it is useful (but not necessary) if the reader is familiar with terms like projective variety, (smooth) curve and genus. Comprehensive sources for this are e.g. the books of Fulton [Ful69] and Hartshorne [Har97].

2.1. Curve equation

Definition 2.1 (Hyperelliptic curve)**.** A *hyperelliptic curve* C over a field k is a smooth projective curve over k with a map $\pi : C \to \mathbb{P}^1$ which is 2-to-1 except for finitely many points where it is 1-to-1.

The map π is called *projection* and it induces a map $\iota : C \to C$ that permutes the preimages of π. The map ι is called *hyperelliptic involution* of C. As a shorthand for a curve defined over k one also writes C/k.

Lemma 2.2. Let k be a field and $\overline{k}$ an algebraic closure of k. Let C/k be a hyperelliptic curve of genus g with a point P defined over k which is invariant under the involution ι. Then we can give an affine model of C by an equation of the form

$$C : y^2 + h(x)y = f(x), \tag{2.1}$$

where $f \in k[x]$ is a monic polynomial of degree $2g+1$, $h \in k[x]$ is a polynomial of degree at most g and no point (x, y) on the curve over $\overline{k}$ simultaneously satisfies both partial derivatives $2y + h(x) = 0$ and $h(x)'y = f'(x)$.

Proof. The proof follows by using the theorem of Riemann-Roch (see Theorem I.5.15 in [Sti93]) and that the dimension of the L-space of $2P$ is $\ell(2P) = 2$. □

This form of the curve is called *Weierstraß form*. In this thesis we consider only curves that satisfy the conditions of Lemma 2.2. Curves of the form (2.1) have exactly one point at infinity, and since they share a lot of properties with imaginary quadratic number fields, they are also called *imaginary hyperelliptic curves*. Aside the imaginary hyperelliptic curves there are also real hyperelliptic curves (see [SSW96] and [JMS04]) which have two points at infinity over $\overline{k}$, but those curves are not of interest in this work. From now on we will use the term "hyperelliptic curve" and mean an "imaginary hyperelliptic curve".

Elliptic curves are covered by Definition 2.1 and can be characterised as curves of genus 1 by Lemma 2.2. Although this is not completely standard, we stick to this since most of the algebraic properties that we are interested in are the same.

The last condition of the lemma ensures that the curve is non-singular, i.e. there are no singular points on C. A singular point is a point on the curve such that both partial derivatives vanish simultaneously. Note that if the characteristic of k is equal to 2, then the polynomial h in (2.1) must be different from 0, otherwise the curve is singular.

From a geometric point of view, a hyperelliptic curve is a smooth (i.e. non-singular), absolutely irreducible, projective variety of dimension 1 with an involution, but considering the affine model of the curve is satisfactory for this work. For more details on varieties and curves we refer to Chapter 4 in [ACD+05] and to the books of Shafarevich [Sha94] and Hartshorne [Har97]. For algebraic curves see also [Ful69].

For a curve C given in the form of (2.1) and an intermediate field L of $\overline{k}/k$, we define $C(L) = \{(x, y) \mid x, y \in L \text{ and } y^2 + h(x)y = f(x)\} \cup \{P_\infty\}$. This is called the set of *L-rational points*. As a shorthand, for $L = \overline{k}$ we write C instead of $C(\overline{k})$.

2.1.1. Examples

Now we give two examples of curves (over the real numbers) having a singular point (also singularity) at the origin. The first type of singularity is called *node* and the second one *cusp*. From the shape one can easily see that the curve has no uniquely defined tangent lines at the point $(0, 0)$ and cannot be a smooth curve.

Example 2.3. Consider the curve given by the equation $y^2 + (x + 1)y = x^5 + 1$ over $\mathbb{F}_5$. This curve is singular because the point $(-1, 0)$ is on the curve and satisfies both partial derivatives and is therefore a singularity. Thus, the equation does not describe a hyperelliptic curve.

Example 2.4. This is an example of a (non-singular) hyperelliptic curve of genus 2 over the real numbers. The curve is given by the equation

$$\begin{aligned} C : y^2 &= x^5 - x^4 - 11x^3 + 9x^2 + 18x \text{ over } \mathbb{R} \\ &= x(x - 2)(x - 3)(x + 1)(x + 3). \end{aligned}$$

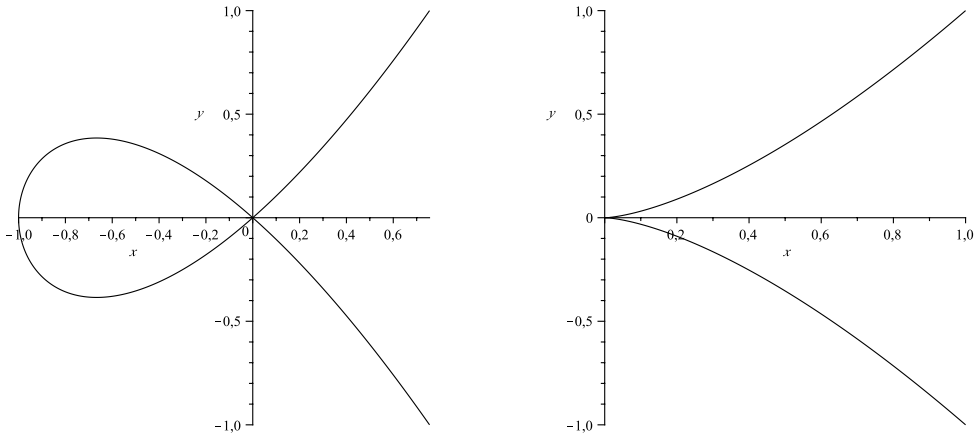


Figure 2.1.: Typical shapes of singular curves with a node (left) and a cusp (right)

The polynomial f on the right-hand side has degree 5 which indicates that the genus is 2. The polynomial h is (by intention) chosen to be 0. We now explain the typical shape of the graph of this hyperelliptic curve.

We first look at points with y-coordinate equal to 0. These are $P_1 = (-3, 0)$, $P_2 = (-1, 0)$, $P_3 = (0, 0)$, $P_4 = (2, 0)$ and $P_5 = (3, 0)$. It is clear that these points satisfy the curve equation because the x-values of the points are exactly the zeros of f. What about points with an x-coordinate between -3 and -1? If we plug in for instance $x = -2$ into the curve equation, we obtain a product of four negative and one positive values the product of which is positive. Since we consider points over the real numbers, we can only extract square roots of positive values. Thus, we get solutions for y only if the right-hand side of the equation is positive or equal to 0, which is the case for points with x-coordinate between -3 and -1, 0 and 2 and greater than 3. So in these ranges, we always have two points with the same x-coordinate, which explains the shape of the curve which is shown in Figure 2.2.

2.2. The function field of a hyperelliptic curve

Now we discuss a very important structure that is related to a hyperelliptic curve, the function field. Via this concept the geometric structure "curve" is associated to the algebraic structure "function field". Interesting properties of a curve can be investigated by looking at its function field. We can associate a (principal) divisor to a function in the function field and thus connect the geometric and the algebraic view. For more details on function fields we refer to the book of Stichtenoth [Sti93]. Now we introduce the coordinate ring of a curve and define its quotient field to be the function field of the curve.

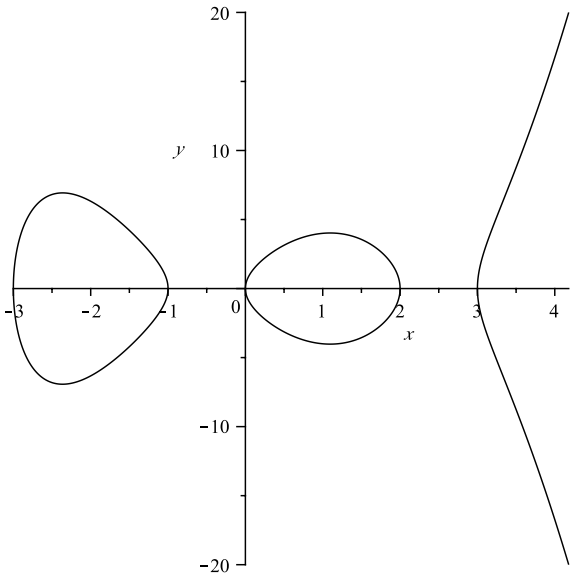


Figure 2.2.: Graph of the hyperelliptic curve C, plotted over the real numbers, with the equation $C : y^2 = x^5 - x^4 - 11x^3 + 9x^2 + 18x$

Definition 2.5 (Coordinate ring, function field)**.** Let the hyperelliptic curve C of genus g be given by the equation $y^2 + h(x)y = f(x)$ over the field k. The *coordinate ring of C over k* is the quotient ring

$$k[C] = k[x, y]/(y^2 + h(x)y - f(x)).$$

Similarly, the coordinate ring of C over $\overline{k}$ is

$$\overline{k}[C] = \overline{k}[x, y]/(y^2 + h(x)y - f(x)).$$

An element of $\overline{k}[C]$ is called *polynomial function on C*. The *function field of C*, denoted by $\overline{k}(C)$, is the field of fractions of the coordinate ring of C. The elements of $\overline{k}(C)$ are called *rational functions on C*.

For a hyperelliptic curve C over k the function field is

$$\overline{k}(C) = \mathrm{Quot}(\overline{k}[x, y]/(y^2 + h(x)y - f(x)). \tag{2.2}$$

This field is equal to the polynomial ring in the variable y over the rational function field $\overline{k}(x)$ (see Example I.1.3 in [Sti93]) modulo the ideal generated by the curve equation, i.e.

$$\overline{k}(x)[y]/(y^2 + h(x)y - f(x)). \tag{2.3}$$

Recall that a hyperelliptic curve is always smooth and absolutely irreducible, i.e. its curve equation $y^2 + h(x)y - f(x)$ is an irreducible polynomial. Hence the ring in (2.3) is a field.

2.3. Algebraic perspective on the function field

In the previous section we have seen that the function field of a hyperelliptic curve is indeed a function field in the sense of Chapter I in [Sti93]. We now explain valuation rings to be able to define the order of a rational function in the function field evaluated at a point on a hyperelliptic curve afterwards.

We first introduce the general basic concepts and identify these in the context of hyperelliptic curves afterwards.

Definition 2.6 (Valuation ring)**.** A *valuation ring* of a field F/k is a ring $\mathcal{O}$ with the following properties:

(1) $k \subsetneqq \mathcal{O} \subsetneqq F$.

(2) For all $r \in F$ we have $r \in \mathcal{O}$ or $r^{-1} \in \mathcal{O}$.

Now we give some important properties of valuation rings. For the proof of the following proposition see [Sti93, Proposition I.1.5 and Theorem I.1.6].

Proposition 2.7 (Properties of valuation rings)**.** Let $\mathcal{O}$ be a valuation ring of a field F/k. Then

(1) $\mathcal{O}$ is a local ring, i.e. it has a unique maximal ideal $\mathcal{P} = \mathcal{O} \setminus \mathcal{O}^*$, where $\mathcal{O}^*$ is the group of units of $\mathcal{O}$.

(2) For $0 \neq r \in F$ we have: $r \in \mathcal{P} \iff r^{-1} \notin \mathcal{O}$.

(3) The unique maximal ideal $\mathcal{P}$ of $\mathcal{O}$ is principal.

(4) The ring $\mathcal{O}$ is a principal ideal domain.

Definition 2.8 (Discrete valuation)**.** A *discrete valuation* of a field F/k is a function $v : F \to \mathbb{Z} \cup \{\infty\}$ with the following properties:

(1) $v(x) = \infty$ if and only if $x = 0$.

(2) $v(xy) = v(x) + v(y)$ for any $x, y \in F$.

(3) $v(x + y) \geq \min\{v(x), v(y)\}$ for any $x, y \in F$.

(4) There exists an element $z \in F$ with $v(z) = 1$.

(5) $v(a) = 0$ for any $0 \neq a \in k$.

2.3.1. Function fields of hyperelliptic curves

Definition 2.9 (Place)**.** A *place* of the function field F/k is a maximal ideal of some valuation ring $\mathcal{O}$ of F/k. The set of all places of F/k is denoted by $\mathbb{P}_F$.

Now we go back to hyperelliptic curves and their function fields. If C is a curve over some field k and $\overline{k}(C)$ its function field, then the set

$$\mathcal{O}_P = \{f \in \overline{k}(C) \mid f = g/h, \text{ where } g, h \in \overline{k}[C] \text{ and } h(P) \neq 0\}$$

is a local ring in $\overline{k}(C)$ with the unique maximal ideal

$$M_P = \{f \in \mathcal{O}_P \mid f(P) = 0\}.$$

If $P \in C$ is a non-singular point, then $\mathcal{O}_P$ is a discrete valuation ring (see [ACD+05, Section 4.4.1]) and thus M_P is a place of the function field $\overline{k}(C)$. The appropriate discrete valuation at P on $\mathcal{O}_P$ is given by

$$\begin{aligned} v_P &: \mathcal{O}_P \to \mathbb{Z} \cup \{\infty\}, \\ v_P(f) &= \max\{i \in \mathbb{Z} \mid f \in M_P^i\}. \end{aligned}$$

We can extend the valuation v_P to the whole function field by defining the valuation as

$$\begin{aligned} v_P &: \overline{k}(C) \to \mathbb{Z} \cup \{\infty\}, \\ v_P(g/h) &= v_P(g) - v_P(h) \quad \text{for } g, h \in \mathcal{O}_P. \end{aligned}$$

With this we are able to define the order of a function evaluated at a point on a hyperelliptic curve. This allows us to define the divisor of a rational function which is essential for the definition of the Picard group of a hyperelliptic curve. See the following two sections for the definition of divisors and principal divisors.

Definition 2.10 (Order of a function at a point)**.** Let C be a curve over a field k, and let be $\overline{k}(C)$ be its function field. For a point $P \in C$ and a rational function $f \in \overline{k}(C)^*$ we define the *order of f at P* as

$$\mathrm{ord}_P(f) = v_P(f).$$

2.4. Divisors and rational functions

We know that the points on an elliptic curve form a group. This is not true for hyperelliptic curves of genus > 1. In the following we will introduce another concept to impose a group structure for hyperelliptic curves, too. Instead of points we use formal sums of points, called divisors, to form a group.

In this section we will introduce divisors, especially degree-0 divisors and principal divisors, which we need to define the divisor class group of a hyperelliptic curve in the next section. This group is of special interest for cryptographers because (for certain curves) the discrete-logarithm problem in it is assumed to be hard (cf. Appendix A).

From now on we assume C to be a hyperelliptic curve of genus g over k. As a shorthand we write C for $C(\overline{k})$.

Definition 2.11 (Divisor, divisor group, degree of a divisor)**.** The free abelian group generated by the points on C is called *divisor group of* C, denoted by $\mathrm{Div}(C)$. The elements of this group are called *divisors*. A divisor $D \in \mathrm{Div}(C)$ is a formal sum of points on C of the form

$$D = \sum_{P \in C} n_P P,$$

where $n_P \in \mathbb{Z}$ and $n_P = 0$ for all but finitely many $P \in C$. The *support* of the divisor D is the set of all points P of D such that $n_p \neq 0$. The sum of two divisors $D_1 = \sum_{P \in C} n_P P$ and $D_2 = \sum_{P \in C} m_P P$ is naturally defined as

$$D_1 + D_2 = \sum_{P \in C} (n_P + m_P) P.$$

The homomorphism

$$\begin{aligned} \deg : \mathrm{Div}(C) &\rightarrow \mathbb{Z} \\ \sum_{P \in C} n_P P &\mapsto \sum_{P \in C} n_P \end{aligned}$$

assigns an integer to each divisor of $\mathrm{Div}(C)$. This integer is called *degree* of the divisor. The set of degree-0 divisors of C is denoted by $\mathrm{Div}^0(C)$ and equals the kernel of $\deg : \mathrm{Div}(C) \rightarrow \mathbb{Z}$. Hence, it is a proper subgroup of $\mathrm{Div}(C)$.

The Galois group $\mathrm{Gal}(\overline{k}/k)$ of the field extension $\overline{k}/k$ acts on the points of C coordinatewise and on $\mathrm{Div}(C)$ and $\mathrm{Div}^0(C)$ via

$$D^\sigma = \sum_{P \in C} n_P P^\sigma$$

for all $\sigma \in \mathrm{Gal}(\overline{k}/k)$ and $D \in \mathrm{Div}(C)$.

Definition 2.12 (Rational divisor)**.** Let L be an intermediate field of $\overline{k}/k$. A divisor D of C is called *L-rational* if

$$D^\sigma = D \quad \text{for all } \sigma \in \mathrm{Gal}(\overline{k}/L).$$

We denote the set of L-rational divisors of C by $\mathrm{Div}_L(C)$ and similarly the set of L-rational divisors of degree 0 by $\mathrm{Div}^0_L(C)$.

Note that for a divisor being L-rational does not necessarily mean that the points in its support are L-rational but the automorphisms in the Galois group do permute the points in the formal sum.

2.5. Principal divisors and divisor class groups

In the last sections we have introduced divisors and rational functions. Now, we combine both notions by introducing the *divisor of a rational function*, which is called *principal divisor*. This, in turn, allows us to define the divisor class group of a hyperelliptic curve. In the next section we will show how the elements of this important group can be represented and explain the group law.

Now we are ready to define the divisor of a rational function, called principal divisor.

Definition 2.13 (Divisor of a rational function, principal divisor)**.** Let $f \in \overline{k}(C)^*$ be a rational function on C.

(1) The *divisor of* f is defined as

$$\mathrm{div}(f) = \sum_{P \in C} \mathrm{ord}_P(f) P.$$

(2) A divisor D is called *principal* if $D = \mathrm{div}(f)$ for some rational function $f \in \overline{k}(C)^*$.

(3) The set of principal divisors on C is denoted by $\mathrm{Princ}(C)$.

Note that $\mathrm{Princ}(C)$ is a group and all principal divisors have degree 0. Hence, $\mathrm{Princ}(C)$ is a subgroup of $\mathrm{Div}^0(C)$.

Definition 2.14 (Divisor class group)**.** The *divisor class group of* C *over* k is the quotient group

$$\mathrm{Pic}^0(C) = \mathrm{Div}^0(C)/\mathrm{Princ}(C).$$

This group is also called *Picard group of* C. For an intermediate field L with $k \subseteq L \subseteq \overline{k}$ we analogously define $\mathrm{Pic}^0_L(C)$ as the group of divisor classes which are invariant under all automorphisms in $\mathrm{Gal}(\overline{k}/L)$. The elements in $\mathrm{Pic}^0(C)$ or $\mathrm{Pic}^0_L(C)$ are called divisor classes or L-rational divisor classes, respectively.

The Picard group is of great importance in cryptography because for certain curves the discrete-logarithm problem in this group is assumed to be hard. Depending on the choice of the curve and the appropriate base field one can show that the Picard group is not vulnerable against currently known attacks (cf. Appendix A). We note that for cryptography we require that the Picard group is finite and thus the underlying fields are finite, but the results in this chapter are correct for general fields.

Note that $\mathrm{Pic}^0_L(C)$ is isomorphic as a group to the L-rational points on the Jacobian of C. So we can identify the divisor classes in the Picard group with the points in the Jacobian variety.

2.5.1. Cardinality of the Picard group

To estimate the number of elements in the Picard group of a hyperelliptic curve, one can use the theorem of Hasse-Weil which gives upper and lower bounds for the number of divisor classes in this group. Observe that the theorem does only depend on the finite field and the genus of the curve.

Theorem 2.15 (Hasse-Weil)**.** The order of the Picard group of a hyperelliptic curve C of genus g over a finite field $\mathbb{F}_q$ is within the range

$$(\sqrt{q}-1)^{2g} \;\leq\; |\mathrm{Pic}^0_{\mathbb{F}_q}(C)| \;\leq\; (\sqrt{q}+1)^{2g}.$$

For more details on the number of points on the curve and the group structure of $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ we refer to Section 5.2 in [ACD+05].

2.6. Arithmetic in divisor class groups

In the previous section we have introduced the divisor class group of a hyperelliptic curve. Now we have to take care of two issues. First, how can an element of a divisor class group be efficiently represented, and second, how can two elements be combined, i.e. how can the group law be performed.

Divisor classes can be represented, using Mumford's theorem, in a unique way by two polynomials of degree less than or equal to the genus of the curve. This provides a very compact representation in the Picard group. The group law can be performed by using Cantor's algorithm [Can87]. We shall present both methods in the following.

2.6.1. Mumford representation

Theorem 2.16 (Mumford)**.** Let C be a hyperelliptic curve of genus g over k. Then each non-trivial k-rational divisor class of C can be represented by a unique pair $[u, v]$ of polynomials $u, v \in k[x]$, where

(1) u is monic,

(2) $\deg(v) < \deg(u) \leq g$,

(3) $u \,|\, v^2 + vh - f$.

Proof. See [ACD+05, Theorem 4.145]. □

This theorem provides a very compact representation of divisor classes and also allows to easily use divisor classes in implementations since only two lists of coefficients (of the two polynomials u and v) of length at most g have to be stored in a computer. In the following we sometimes write a divisor class as such

a list because explicit addition, doubling or halving formulas work directly on the coefficients. It will be always clear from the context whether we work with coefficients or polynomials.

In the Mumford representation the first polynomial $u(x)$ splits over $\overline{k}$ into (at most) g linear factors $x - a_i$, where the values of the a_i are the x-coordinates of the affine points of the representative of the divisor class. So the roots of $u(x)$ are exactly the x-coordinates of the points of the divisor class. The polynomial $v(x)$ is a function that maps the x-coordinate of each point to its y-coordinate.

Since the degree of $u(x)$ is less than or equal to the g, Theorem 2.16 has the important consequence that each divisor class of a hyperelliptic curve of genus g can be represented by a divisor, where the degree of u is at most g.

Example 2.17. We consider the genus-2 hyperelliptic curve given by

$$C : y^2 + xy = x^5 + 3x^3 + 5x + 1 \tag{2.4}$$

over $\mathbb{F}_{17}$. The points $P_1 = (15, 16)$ and $P_2 = (5, 9)$ lie on the curve and form the divisor $D = P_1 + P_2 - 2P_\infty$. The divisor class $\overline{D}$ of D in Mumford form is represented by $\overline{D} = [x^2 + 14x + 7, 16x + 14]$. To see this we check the conditions of the above theorem. The two first conditions are obviously satisfied. And the last one also holds true since

$$v^2 + vh - f = 16x^5 + 14x^3 + 15x + 8 = (x^2 + 14x + 7)(x^3 + 3x^2 + 5x + 11).$$

2.6.2. The group law

After introducing the representation of the elements in the Picard group, we have to care about how to carry out the group law. For that, Cantor's algorithm [Can87] can be used. This algorithm allows the addition of two divisor classes in the Picard group of the curve. If one wants to add a divisor class to itself, then this is called *divisor class doubling*.

Looking at Cantor's algorithm we see that it consists mainly of two parts. First, the two divisor classes are combined (combination step) and second, the intermediate result is reduced (reduction step) to obtain the result in Mumford representation.

Algorithm 1 (Cantor)

INPUT:	Two divisor classes $\overline{D}_1 = [u_1, v_1]$ and $\overline{D}_2 = [u_2, v_2]$ on the curve $C : y^2 + h(x)y = f(x)$.
OUTPUT:	The unique reduced divisor D such that $\overline{D} = \overline{D}_1 \oplus \overline{D}_2$.

1: $d_1 \leftarrow \gcd(u_1, u_2)$ ▷ $[d_1 = e_1u_1 + e_2u_2]$

2: $d \leftarrow \gcd(d_1, v_1 + v_2 + h)$ ▷ $[d = c_1d_1 + c_2(v_1 + v_2 + h)]$

3: $s_1 \leftarrow c_1 e_1,\ s_2 \leftarrow c_1 e_2,\ s_3 \leftarrow c_2$
4: $u \leftarrow \frac{u_1 u_2}{d^2},\ v \leftarrow \frac{s_1 u_1 u_2 + s_2 u_2 v_1 + s_3(v_1 v_2 + f)}{d} \pmod{u}$
5: **while** $\deg(u) > g$
6: $\quad u' \leftarrow \frac{f - vh - v^2}{u},\ v' \leftarrow (-h - v) \pmod{u'}$
7: $\quad u \leftarrow u',\ v \leftarrow v'$
8: **end while**
9: make u monic
10: **return** $[u, v]$

Remark 2.18. Note that Cantor's algorithm is completely general and holds for any genus and any field. As we will see in Chapters 3 and 4, we obtain explicit formulas from this algorithm by restricting it to binary fields and to a certain class of hyperelliptic curves.

We will consider Cantor's algorithm in two specialised versions (genus 2 and genus 3) to get best performance on the one hand and to get the possibility to invert its steps in order to obtain halving formulas on the other hand. For the explicit formulas see Chapters 3 and 4.

Note that the loop in Steps 5 to 8 will terminate when $\deg(u)$ is less than or equal to the genus of the curve, i.e. the sum of $\overline{D}_1$ and $\overline{D}_2$ is represented by at most g points on the curve.

Doubling, halving and scalar multiplication

Cantor's algorithm can compute the sum of two divisor classes in the Picard group. This is called *divisor class addition*, denoted by $\overline{D}_1 \oplus \overline{D}_2$. We use a special symbol for the addition here in order to not confuse it with the addition in the underlying field. If $\overline{D}_1 = \overline{D}_2$, then we speak of *divisor class doubling*. One can still write $\overline{D}_1 \oplus \overline{D}_1$ but we use $[2]\overline{D}_1$ to denote the doubling of $\overline{D}_1$.

In the Picard group it is always possible to compute the double of a divisor class. This can be seen as a mapping that maps a divisor class $\overline{D}$ to its double $[2]\overline{D}$. In certain situations it is also possible to give the inverse map of the doubling which is called *divisor class halving*. In Chapters 3 and 4 we investigate curves of genus 2 and 3 which allow us to define the halving map. To be more precise we define divisor class halving as follows: Given a divisor class $\overline{E} = [2]\overline{D}$ we want to find the divisor class $\overline{D}$. For the halving we also write informally $[\frac{1}{2}]\overline{E} = \overline{D}$.

The doubling map $\overline{D} \mapsto [2]\overline{D}$ can be generalised to $\overline{D} \mapsto [n]\overline{D}$ for an arbitrary integer n, as the next definition shows.

Definition 2.19 (Multiplication-by-n map, torsion element)**.** Let n be an integer

and let $\overline{D} \in \mathrm{Pic}^0(C)$ be a divisor class of the curve C. The map

$$\begin{aligned} [n] : \mathrm{Pic}^0(C) &\to \mathrm{Pic}^0(C) \\ \overline{D} &\mapsto [n]\overline{D} = \underbrace{\overline{D} \oplus \ldots \oplus \overline{D}}_{n\text{-times}} \end{aligned}$$

is called *multiplication-by-n map* (also: scalar multiplication) on the Picard group of C. The kernel of this mapping $[n]$ is denoted $\mathrm{Pic}^0(C)[n]$, and an element in $\mathrm{Pic}^0(C)[n]$ is called *n-torsion element*.

2.7. Torsion points and p-rank

A very important invariant of a hyperelliptic curve C over a finite field $\mathbb{F}_{p^k}$ is the p-rank. For instance, in Chapter 3 we classify binary hyperelliptic curves of genus 2 into three categories depending on their 2-rank. We will show that curves with 2-rank 1 have noticeable advantages over the others.

For cryptosystems based on elliptic or hyperelliptic curves the most important operation is scalar multiplication (see Definition 2.19). This is the n-fold multiplication of a divisor class on a hyperelliptic curve to itself. Given an integer n and a divisor class $\overline{D}$ we would like to compute $[n]\overline{D} = \overline{D} \oplus \ldots \oplus \overline{D}$. This is almost always done by using a double-and-add like algorithm. For more details on this we refer to Chapter 9 in [ACD+05].

The following theorem and definition are taken from Section 14.1.4 in [ACD+05].

Theorem 2.20. Let C be a hyperelliptic curve defined over a field k and let n be an integer. If the characteristic of k is either 0 or prime to n, then

$$\mathrm{Pic}^0(C)[n] \cong (\mathbb{Z}/n\mathbb{Z})^{2g}.$$

When $\mathrm{char}(k) = p$ there exist an integer r such that

$$\mathrm{Pic}^0(C)[p^e] \cong (\mathbb{Z}/p^e\mathbb{Z})^r,$$

where $0 \leq r \leq g$ and r is the same for all $e \geq 1$.

Definition 2.21. Let k be a field of characteristic p and let C be a hyperelliptic curve defined over k. The *p-rank of C over k* is defined to be the integer r in Theorem 2.20.

Note that the p-rank of a hyperelliptic curve C is always less than or equal to its genus.

The next proposition characterises the structure of the Picard group of a curve of arbitrary genus over a finite field $\mathbb{F}_q$.

Proposition 2.22. Let $C/\mathbb{F}_q$ be a curve of genus g. For the structure of the group of the $\mathbb{F}_q$-rational elements in the Picard group of C we have

$$\mathrm{Pic}^0_{\mathbb{F}_q}(C)[n] \cong \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z} \times \ldots \times \mathbb{Z}/n_{2g}\mathbb{Z}, \tag{2.5}$$

where $n_i \,|\, n_{i+1}$ for $1 \leq i < 2g$, and for all $1 \leq i \leq g$ one has $n_i \,|\, q - 1$.

Supersingular curves

An elliptic curve E is called *supersingular* if it has p-rank 0. The Jacobian variety (or Picard group) of a hyperelliptic curve is called supersingular if it is the product of supersingular elliptic curves. Thus, the p-rank of a supersingular Jacobian variety is 0, but the converse does not have to be true. Usually we speak of a supersingular hyperelliptic curve if its Jacobian variety is supersingular.

Supersingular curves always have a small embedding degree (see Section A.5.1), i.e. the Weil and Tate pairings map to a small extension field of $\mathbb{F}_q$. For cryptographic applications using the DLP, this is considered a weakness (see the section on the MOV and Frey-Rück attack in Appendix A.5). In Chapters 3 and 4 we classify binary hyperelliptic curves of genus 2 and 3 depending on their 2-rank and identify supersingular curves.

3. Arithmetic on Genus-2 Curves over Binary Fields

In this chapter, we investigate the arithmetic in the divisor class group of hyperelliptic curves of genus 2 over binary fields. Our focus is on the efficiency of the arithmetic on these curves. First, we classify curves of genus 2 depending on their 2-rank and give explicit addition, doubling and halving formulas to be able to perform scalar multiplication in the divisor class group of the curve. We show that (for certain classes of curves) the scalar multiplication using a halve-and-add algorithm can be faster than the traditional double-and-add method. Point halving on elliptic curves proved already successful (see [Sch00b]).

For the different types of curves we provide explicit doubling and halving formulas for fields of characteristic 2. We also give explicit addition formulas which can be used in arbitrary characteristic. The last part of the chapter contains inversion-free doubling and addition formulas, which are useful when the computation of inverse elements in the base field is rather costly.

The doubling formulas in Section 3.3 are taken from the paper by Lange and Stevens [LS05], and can be found along with the addition formulas in Section 14.5 in [ACD+05].

The new contributions of this chapter are an extended classification of hyperelliptic curves of genus 2 over binary fields, going beyond [CY02] and [LS05]; a complete study of explicit halving formulas for all curves with a Picard group of order $2r$, where r is odd; inversion-free addition and doubling formulas. The halving formulas improve our own result in [Bir07] and the previous ones by Kitamura, Katagi and Takagi [KKT05]. This and the classification is joint work with Nicolas Thériault and has been published in [BT08]. The inversion-free formulas are joint work with Tanja Lange.

3.1. Classification of genus-2 curves

A hyperelliptic curve C of genus 2 over a binary field k can be given by an equation of the form

$$C : y^2 + h(x)y = f(x), \tag{3.1}$$

where $h(x) = h_2x^2 + h_1x + h_0 \neq 0$ and $f(x) = f_5x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$ are polynomials over k. As stated in Definition 2.1, the polynomial $f \in k[x]$ has degree $5 = 2g + 1$, where g is the genus of the curve. The non-zero polynomial h

is of degree at most $g = 2$, and no point on the curve C over the algebraic closure $\overline{k}$ satisfies both partial derivatives of (3.1). Recall that in case of characteristic 2 we require that $h \neq 0$, otherwise there are singular points.

It is customary to use curve isomorphisms to impose that f is monic, but we will relax this condition for some curves as the halving formulas are more efficient if we use the isomorphisms to have a monic polynomial h at the cost of a non-monic polynomial f.

In this chapter, the field k is binary, i.e. $k = \mathbb{F}_{2^d}$ for some $d > 0$. To a priori eliminate Weil descent attacks (see Appendix A.6 or [GHS02]) we require the extension degree d to be prime. There are also some prime extension degrees which should be avoided, e.g. Mersenne and Fermat primes. We note that this implies that d is odd, and so the mappings $\alpha \mapsto \alpha^3$ and $\alpha \mapsto \alpha^5$ are isomorphisms in $\mathbb{F}_{2^d}$. Therefore, we can extract third and fifth roots in $\mathbb{F}_{2^d}$ (see the isomorphisms for Types II and III in Section 3.1.2).

3.1.1. The 2-rank

Roughly speaking, the 2-rank (see Definition 2.21) of a hyperelliptic curve C is the number of copies of $\mathbb{Z}/2\mathbb{Z}$ in the 2-torsion group of the divisor class group of C. The 2-rank is less than or equal to the genus of C, hence it is at most 2 in the present situation. The next lemma relates the degree of the polynomial h in (3.1) to the 2-rank (see Definition 2.21) of C and gives a condition when it is supersingular (see Section 2.7).

Lemma 3.1. Let C be a hyperelliptic curve of genus 2 over a binary field k, given by an equation of the form $y^2 + h(x)y = f(x)$ where $h \neq 0$. Then the following is true:

(1) If $\deg(h) = 0$, then the 2-rank of C is 0 and C is supersingular.

(2) If $\deg(h) = 1$, then the 2-rank of C is equal to 1 and C is non-supersingular.

(3) If $\deg(h) = 2$, then the 2-rank of C is equal to 1 or 2 and C is non-supersingular.

Proof. In (1), to see that C is supersingular if it has 2-rank 0, we refer to [Gal01, Theorem 4]. To compute the 2-rank of C we have to look at the 2-torsion group $\mathrm{Pic}^0(C)[2]$ of the Picard group of C. A divisor class $\overline{D} = [u, v]$ in $\mathrm{Pic}^0(C)$ is of 2-torsion if $2\overline{D} = [1, 0]$ which is equivalent to $\overline{D} = -\overline{D}$, i.e. $[u, v] = [u, -v - h]$ which is equal to $[u, v + h]$ as k has characteristic 2. We now look for divisor classes for which $v \equiv v + h \pmod{u}$ holds.

(1) In the first case we have $\deg(h) = 0$, which means $h = c$ for a non-zero constant $c \in k$ and therefore of course $v \neq v + c$. Thus, there is only the trivial divisor class $\overline{E} = [1, 0]$ satisfying $2\overline{E} = [1, 0]$. It follows that the 2-rank of C equals 0.

(2) Here we have $\deg(h) = 1$ and h has exactly one root x_0 over k. The equation $v = v + h$ is only true for $x = x_0$. Thus, there are two divisor classes $\overline{D}$ satisfying $2\overline{D} = [1, 0]$, namely the trivial one and $\overline{D} = [u, v]$ where $u = x - x_0$ and $v = \sqrt{f(x_0)}$. This can be seen by plugging in x_0 for x. We get $v(x_0) = \sqrt{f(x_0)} = v(x_0) + h(x_0)$.

It is clear that $\overline{D} = [x - x_0, \sqrt{f(x_0)}]$ is a valid divisor class because it comes from the degree-0 divisor $P - P_\infty$ where $P = (x_0, \sqrt{f(x_0)})$ is a point on C.

(3) In this case we have $\deg(h) = 2$ and h has two roots x_1 and x_2 over $\overline{k}$. We have to distinguish the two cases: $x_1 = x_2$ and $x_1 \neq x_2$. In the first case the 2-rank equals 1 since $h(x) = 0$ only for $x = x_1$. Thus, we have $\mathrm{Pic}^0(C)[2] = \{[1, 0], \overline{D}_1\} \cong \mathbb{Z}/2\mathbb{Z}$, where $\overline{D}_1 = [x - x_1, \sqrt{f(x_1)}]$.

In the second case, where $x_1 \neq x_2$, the divisor classes $[1, 0], \overline{D}_1 = [x - x_1, \sqrt{f(x_1)}]$ and $\overline{D}_2 = [x - x_2, \sqrt{f(x_2)}]$ are of 2-torsion for the same reason as in (2). Only one more class of 2-torsion is possible, namely

$$\overline{D}_3 = \left[(x - x_1)(x - x_2), \frac{y_2 - y_1}{x_2 - x_1}x + \frac{y_2 - y_1}{x_2 - x_1}x_1 + y_1\right].$$

This class comes from the divisor $P_1 + P_2 - [2]P_\infty$ where $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ are points on C and x_1, x_2 roots of h. Since the 2-rank is at most 2, we have $\mathrm{Pic}^0(C)[2] = \{[1, 0], \overline{D}_1, \overline{D}_2, \overline{D}_3\}$. Under the group operation via Cantor's algorithm, the maximal order of each element is 2. Thus $\mathrm{Pic}^0(C)[2] \cong \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ and therefore the 2-rank is 2. □

Observe that hyperelliptic curves with 2-rank 0 are supersingular (see Section 2.7) and therefore not suitable for DLP-based cryptography because they are vulnerable against Frey-Rück attacks (cf. Appendix A.5 and [FR94]). Galbraith [Gal01, Section 4] has found a criterion to check whether a hyperelliptic curve is supersingular or not. His result is even more general and covers general abelian varieties over finite fields. Although interest in supersingular curves is generally limited to pairing based cryptography, we will still cover these curves for the sake of completeness.

3.1.2. Classification and isomorphic transformations

In this section, we impose a classification of hyperelliptic curves of genus 2 over binary fields. Depending on the degree of $h(x)$ the curves are sorted into three main types: I, II and III. For curves of Type I (i.e. $\deg(h) = 2$) we have three subtypes: Ia, Ib and Ic. Here we sort the curves depending on the number of k-rational roots of $h(x)$.

In the following, we show for each type of curve how to apply isomorphic transformations to the curve equation in order to simplify it as much as possible

while not losing any generality. In this context, we need to compute the (absolute) trace of an element in the finite field.

Definition 3.2 (Trace)**.** Let α be an element of the finite field $\mathbb{F}_{q^k}$. The *trace* of α is given by the formula

$$\mathrm{TR}_{\mathbb{F}_{q^k}/\mathbb{F}_q}(\alpha) = \prod_{i=1}^{k} \alpha^{q^i}. \tag{3.2}$$

If it is clear from the context which field we are working with, then we will write $\mathrm{TR}(\alpha)$ instead of $\mathrm{TR}_{\mathbb{F}_{q^k}/\mathbb{F}_q}(\alpha)$. See also Definition 2.52 and Proposition 2.97 in [ACD+05].

Lemma 3.3. For a finite field $\mathbb{F}_{q^k}$ the trace function $\mathrm{TR}_{\mathbb{F}_{q^k}/\mathbb{F}_q}$ satisfies the following properties:

(1) $\mathrm{TR}_{\mathbb{F}_{q^k}/\mathbb{F}_q}(\alpha+\beta) = \mathrm{TR}_{\mathbb{F}_{q^k}/\mathbb{F}_q}(\alpha) + \mathrm{TR}_{\mathbb{F}_{q^k}/\mathbb{F}_q}(\beta)$ for all $\alpha, \beta \in \mathbb{F}_{q^k}$.

(2) $\mathrm{TR}_{\mathbb{F}_{q^k}/\mathbb{F}_q}(c\alpha) = c \cdot \mathrm{TR}_{\mathbb{F}_{q^k}/\mathbb{F}_q}(\alpha)$ for all $c \in \mathbb{F}_q$ and $\alpha \in \mathbb{F}_{q^k}$.

(3) $\mathrm{TR}_{\mathbb{F}_{q^k}/\mathbb{F}_q}(\alpha^q) = \mathrm{TR}_{\mathbb{F}_{q^k}/\mathbb{F}_q}(\alpha)$ for all $\alpha \in \mathbb{F}_{q^k}$.

Proof. See Theorem 2.23 in [LN97]. □

We also require to solve quadratic equations over $\mathbb{F}_{2^d}$. We point out that the equation $x^2+x+c = 0$ has two solutions in $\mathbb{F}_{2^d}$ precisely if $\mathrm{TR}(c) = 0$. To compute the solutions, we need to calculate the *half-trace* of a field element (see Section 11.2.6 in [ACD+05] for a description). We denote the half-trace function by HT. We note that the computation of solutions of an quadratic equation depends on the extension degree d being odd or even. In this work we will have odd values for d only.

For a curve C given by (3.1), the possible isomorphisms are

$$x \mapsto \alpha x + \beta \quad \text{and} \quad y \mapsto \gamma y + \delta x^2 + \varepsilon x + \zeta, \tag{3.3}$$

where both α and γ are non-zero. After applying the isomorphisms, we need to divide the curve equation by γ^2 to make it monic. In this way, we arrive at the following five types of curves:

(Ia) $\deg(h) = 2$ (i.e. $h_2 \neq 0$) and $h(x)$ irreducible over k:

We first note that $h_1 \neq 0$ (otherwise $h(x)$ would be a square) and $h_0 \neq 0$ (otherwise $h(x)$ would be reducible). For the halving formulas it will be better to have $h_1 = 1$ than $f_5 = 1$. So the first step is to force $h_2 = h_1 = 1$. Applying the maps $x \mapsto \alpha x$ with $\alpha = h_1/h_2$ and $y \mapsto \gamma y$ with $\gamma = h_1^2/h_2$ and dividing the equation by γ^2 afterwards, yields an equation of the form

$$y^2 + (x^2 + x + h_0)y = f_5x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0. \tag{3.4}$$

Since h is irreducible over k, the trace of h_0 is 1 (otherwise $h(x)$ would split). Applying the transformation $x \mapsto x + \beta$ with $\beta = \mathrm{HT}(h_0 + 1)$, we get $\beta^2 + \beta + h_0 + 1 = 0$, and we can replace $\beta^2 + \beta + h_0$ by 1. Hence $h(x)$ can be transformed into $x^2 + x + 1$.

We would also like to have $\mathrm{TR}(f_4) \cdot \mathrm{TR}(f_5) = 0$. Only if $\mathrm{TR}(f_4) = \mathrm{TR}(f_5) = 1$, this is not satisfied. The transformation $x \mapsto x + 1$ replaces f_4 by $f_4 + f_5$ and does neither change f_5 nor $h(x)$. After this transformation, the product of the traces is 0. Hence we can assume to always have $\mathrm{TR}(f_4) \cdot \mathrm{TR}(f_5) = 0$.

The next step is to apply the map $y \mapsto y + \delta x^2$. With this, the coefficient of x^4 becomes $\delta^2 + \delta + f_4$. With $\delta = \mathrm{HT}(f_4)$, we see that the coefficient of x^4 is either 0 or 1.

Applying the maps $y \mapsto y + f_3 x$ and $y \mapsto y + f_2$ forces $f_3 = f_2 = 0$ and the curve equation is of the form

$$y^2 + (x^2 + x + 1)y = f_5 x^5 + f_4 x^4 + f_1 x + f_0, \tag{3.5}$$

where $f_4 \in \mathbb{F}_2$ and $f_4 \cdot \mathrm{TR}(f_5) = 0$.

(Ib) $\deg(h) = 2$ (i.e. $h_2 \neq 0$) and $h(x)$ is the product of two distinct linear factors:

Note that $h_1 \neq 0$ (otherwise $h(x)$ would be a square). Using β and one of the roots of $h(x)$, we can obtain $h_0 = 0$ via the map $x \mapsto x + \beta$. After that, we can use α and γ to restrict $h(x)$ to $x^2 + x$. As in the previous case, we can also impose $\mathrm{TR}(f_5) \cdot \mathrm{TR}(f_4) = 0$. Taking advantage of δ, we can restrict f_4 to $\mathbb{F}_2$. Afterwards, using ε and ζ allows us to remove f_3 and f_2. So the curve equation has the form

$$y^2 + (x^2 + x)y = f_5 x^5 + f_4 x^4 + f_1 x + f_0, \tag{3.6}$$

where $f_4 \in \mathbb{F}_2$ and $f_4 \cdot \mathrm{TR}(f_5) = 0$.

(Ic) $\deg(h) = 2$ (i.e. $h_2 \neq 0$) and $h(x)$ is a square:

Observe that $h_1 \neq 0$ (otherwise $h(x)$ would not be a square). Using α, β and γ we can force $h(x) = x^2$ and make $f(x)$ monic. With ε and ζ we can remove f_3 and f_2. Finally, δ can be used to limit f_4 to $\mathbb{F}_2$. Hence a curve of Type Ic has an equation of the form

$$y^2 + x^2 y = x^5 + f_4 x^4 + f_1 x + f_0, \tag{3.7}$$

where $f_4 \in \mathbb{F}_2$.

(II) $\deg(h) = 1$ (i.e. $h_2 = 0$, $h_1 \neq 0$):

Applying the isomorphisms with $\alpha = (h_1^2/f_5)^{1/3}$, $\beta = h_0/h_1$ and $\gamma = h_1\alpha$, we obtain $h(x) = x$ and make $f(x)$ monic. Using δ and ζ allows us to

remove f_4 and f_1. After that, if we apply the transformation $y \mapsto y + \varepsilon x$, then the coefficient of x^2 equals $\varepsilon^2 + \varepsilon + f_2$, and we can replace f_2 by the trace of f_2. Thus the curve equation is of the form

$$y^2 + xy = x^5 + f_3 x^3 + f_2 x^2 + f_0, \tag{3.8}$$

where $f_2 \in \mathbb{F}_2$.

(III) $\deg(h) = 0$ (i.e. $h_2 = h_1 = 0$):

Using $\alpha = (h_0^2/f_5)^{1/5}$ and $\gamma = h_0$, we can force $h(x) = 1$ and make $f(x)$ monic. With $\delta = \sqrt{f_4}$ and $\varepsilon = \sqrt{f_2}$ we can remove f_4 and f_2. Finally, $y \mapsto y + \zeta$ can be used to limit f_0 to $\mathbb{F}_2$, since we can replace $\zeta^2 + \zeta + f_0$ by 0 or 1. The curve equation is of the form

$$y^2 + y = x^5 + f_3 x^3 + f_1 x + f_0, \tag{3.9}$$

where $f_0 \in \mathbb{F}_2$.

Note that we did not include the non-singularity condition, nor conditions on the group order in the descriptions of the different types. In terms of isomorphism classes, Types Ia and Ib are the most common (each with $\frac{3}{2}q^3 + O(q^2)$ different classes), followed by Types II and Ic (each with $2q^2 + O(q)$ classes) and with Type III (supersingular) the least common ($O(q)$ classes). For more details on the number of isomorphism classes of hyperelliptic curves of genus 2, we refer the reader to [CY02]. Section 3 of this paper treats the case of binary fields.

3.1.3. Conditions on the order of the Picard group

In the following study, we limit ourselves to curves for which the order of $\mathrm{Pic}^0_{\mathbb{F}_{2^d}}(C)$ is either odd (i.e. $h(x)$ is constant), or $2r$ for an odd number r (which eliminates all curves of Type Ib). This restriction is necessary to get a better performance out of the halving. For any hyperelliptic curve, the halve-and-add algorithm allows us to compute a scalar multiple of a divisor class if it is contained in a subgroup of odd order. In this way, the preimage of the doubling can always be computed and "becomes" unique (all other preimages of the doubling have even order). The group order conditions are due to the following reasons:

(1) To verify that the preimage is in the subgroup of odd order, we make sure that it can be halved again as many times as we want. If the group contains divisor classes of order 2^r, then we use as a test criterion that we can halve the preimage (at least) r times, which obviously affects the cost of our halving formulas. When $r \geq 2$ (e.g. when there is a divisor class of order 4), the increased work required for this check becomes too expensive for the halving to be interesting.

(2) If C is of Type Ib, then there are four possible preimages of the doubling map. The halving formula must then distinguish which of the four is in the subgroup of odd order, which significantly increases the cost of the halving. We also computed formulas in this case, and the halving does indeed become much more expensive than the doubling.

When we consider all the isomorphism classes for a given type of curve (other than Type III), between a half and two thirds of them have divisor classes of order 4, so rejecting these curves has an acceptably small impact on the number of possible curves. Furthermore, because of the attack of Pohlig and Hellman (see Appendix A.1 and [PH78]), curves with a divisor class of order 4 are slightly weaker than those with one of order 2 only. So the restriction can be seen as advantageous for the security of the curves. From a cryptographic perspective, the two most interesting types of curves for halving formulas are Type II (most efficient halving) and Type Ia (largest number of isomorphism classes). In terms of the benefits of halving over doubling, Type Ia gives the best savings, mostly because Type II has very efficient doubling.

3.1.4. Notation

As always, we write divisor classes in Mumford representation, i.e. we use the form $\overline{D} = [u_a, v_a]$, where u_a and v_a are polynomials satisfying the conditions of Theorem 2.16. We will use u_a and v_a for the inputs and u_c and v_c for the outputs of our algorithms. Accordingly, the coefficients are denoted by u_{ai}, v_{ai}, u_{ci} and v_{ci} for $i = 0, 1, 2, \ldots$

Furthermore, in the following algorithms we denote a field multiplication by M, a field inversion by I, a squaring by S and the extraction of a square root by SR. For a half-trace computation we write HT, for a trace computation TR (cf. Section 3.1.2).

3.2. Addition for arbitrary characteristic

When a scalar multiple of a divisor class is computed using a double-and-add like algorithm, a doubling is computed for each bit of the scalar; an addition is only computed for each non-zero bit. Nevertheless, the addition is important and we need explicit formulas. The following algorithm for divisor class addition is taken from [ACD$^+$05, Algorithm 14.19] and works for arbitrary characteristic.

Note that if the characteristic is different from 5, then we can always achieve $h_2 \in \mathbb{F}_2$ and $f_4 = 0$ by isomorphic transformations. Thus, we do not include multiplications by h_2 and f_4 in the operation count. For curves of Types Ia and Ib slightly different formulas will be necessary if $f_5 \neq 1$.

Algorithm 2 (Addition for genus-2 curves, $\deg(u_1) = \deg(u_2) = 2$)

INPUT: Two divisor classes $[u_1, v_1], [u_2, v_2]$ with $u_i = x^2 + u_{i1}x + u_{i0}$ and $v_i = v_{i1}x + v_{i0}$

OUTPUT: $[u', v'] = [u_1, v_1] \oplus [u_2, v_2]$

1: $z_1 \leftarrow u_{11} - u_{21}$, $z_2 \leftarrow u_{20} - u_{10}$, $z_3 \leftarrow u_{11}z_1 + z_2$ ▷ 1M
2: $r \leftarrow z_2z_3 + z_1^2u_{10}$, $w_0 \leftarrow v_{10} - v_{20}$, $w_1 \leftarrow v_{11} - v_{21}$ ▷ 2M+1S
3: $w_2 \leftarrow z_3w_0$, $w_3 \leftarrow z_1w_1$ ▷ 2M
4: $s'_1 \leftarrow (z_1 + z_3)(w_0 + w_1) - w_2 - w_3(1 + u_{11})$ ▷ 2M
5: $s'_0 \leftarrow w_2 - u_{10}w_3$ ▷ 1M
6: **if** $s'_1 = 0$ **then**
7: $s_0 \leftarrow s'_0r^{-1}$, $u'_0 \leftarrow f_4 - u_{21} - u_{11} - s_0^2 - s_0h_2$ ▷ 1I+1M+1S
8: $w_1 \leftarrow s_0(u_{21} - u'_0) + h_1 + v_{21} - h_2u'_0$ ▷ 1M
9: $w_2 \leftarrow u_{20}s_0 + v_{20} + h_0$, $v'_0 \leftarrow u'_0w_1 - w_2$ ▷ 2M
10: **else**
11: $w_1 \leftarrow (rs'_1)^{-1}$, $w_2 \leftarrow rw_1$, $w_3 \leftarrow s_1'^2w_1$ ▷ 1I+3M+1S
12: $w_4 \leftarrow rw_2$, $w_5 \leftarrow w_4^2$, $s''_0 \leftarrow s'_0w_2$ ▷ 2M+1S
13: $l'_2 \leftarrow u_{21} + s''_0$, $l'_1 \leftarrow u_{21}s''_0 + u_{20}$, $l'_0 \leftarrow u_{20}s''_0$ ▷ 2M
14: $u'_0 \leftarrow (s''_0 - u_{11})(s''_0 - z_1 + h_2w_4) - u_{10}$ ▷ 1M
15: $u'_0 \leftarrow u'_0 + l'_1 + (h_1 + 2v_{21})w_4 + (2u_{21}z_1 - f_4)w_5$ ▷ 2M
16: $u'_1 \leftarrow 2s''_0 - z_1 + h_2w_4 - w_5$
17: $w_1 \leftarrow l'_2 - u'_1$, $w_2 \leftarrow u'_1w_1 + u'_0 - l'_1$ ▷ 1M
18: $v'_1 \leftarrow w_2w_3 - v_{21} - h_1 + h_2u'_1$ ▷ 1M
19: $w_2 \leftarrow u'_0w_1 - l'_0$, $v'_0 \leftarrow w_2w_3 - v_{20} - h_0 + h_2u'_0$ ▷ 2M
20: **end if**
21: **return** $[u', v']$ ▷ Total: 1I+22M+3S (1I+12M+2S if $s'_1 = 0$)

3.3. Doubling

In this section we give explicit doubling formulas for hyperelliptic curves of Types Ia, Ib, Ic and II. The formulas were derived from Cantor's algorithm (see Section 2.6.2) by restricting it to genus 2 and to fields of characteristic 2.

3.3.1. Distinguishing the cases

In genus 2, if we are given a non-trivial divisor class $\overline{D} = [u_a, v_a]$ in Mumford representation, then the degree of the polynomial u_a is either 1 or 2. We will now discuss divisor class doubling with Cantor's algorithm for both cases and investigate for each case the possible degree of u_c in $[u_c, v_c] = [2]\overline{D}$. We start with a divisor class $\overline{D} = [u_a, v_a]$.

(1) Let us assume $\deg(u_a) = 1$. In Step 4 of Algorithm 1 (Cantor), we compute $u = u_a^2$ and $v = c_1 u_a v_a + c_2(v_a^2 + f) \pmod{u}$, where $c_1 = u_1^{-1} \pmod{h}$ and $c_2 = h^{-1} \pmod{u_1}$. The degree of u is 2, and thus the degree of v is less than 2. The algorithm of Cantor now stops because $\deg(u) \leq 2$. We denote this case by DBL12, since we doubled a divisor class with $\deg(u_a) = 1$ and the degree of the output u_c is 2.

(2) If $\deg(u_a) = 2$, then in Step 4 of Algorithm 1, we compute $u = u_a^2$ and $v = c_1 u_a v_a + c_2(v_a^2 + f) \pmod{u}$. The degree of u is 4, and thus the degree of v is less than 4. Next, we get into the loop in Steps 5 to 8, where u and v are reduced. In Step 6, we compute $u' = (f + vh + v^2)/u$ and $v' = h + v \pmod{u'}$. Now, the degree of u' is determined by $5 \leq \deg(f + v^2) \leq 6$.

If $\deg(f + v^2) = 5$, then $\deg(u') = \deg(f) - \deg(u) = 5 - 4 = 1$ and Cantor's algorithm stops. We denote this case by DBL21.

The other case is $\deg(f + v^2) = 6$. In this case, the degree of u' is $\deg(v^2) - \deg(u) = 6 - 4 = 2$ and the algorithm stops. We denote this case by DBL22.

By degree reasons, there are no other possible cases and we can summarise that in genus 2 we have the following three doubling cases: DBL12, DBL21 and DBL22 (There are also the trivial cases DBL10 and DBL20, but we do not consider them here separately). Necessarily, we have the following halving cases: HLV21, HLV12 and HLV22 (if the halving map can be defined).

3.3.2. Type Ia: $h(x) = x^2 + h_1 x + h_0$

In the next algorithm which is taken from [LS05, Table 3] we give formulas to double a divisor class on a curve with the equation

$$C : y^2 + (x^2 + h_1 x + h_0)y = x^5 + f_1 x + f_0. \tag{3.10}$$

This form of the curve equation is more general and covers curves of Type Ia for $h_1 = h_0 = 1$. In this case the operations count of the doubling algorithm drops down to 1I+15M+7S.

The algorithm computes the double of a divisor class $\overline{D} = [u_a, v_a]$, where the degree of the polynomial u_a equals 2. We assume that the polynomial u_c in the output divisor class $[u_c, v_c] = [2]\overline{D}$ has degree 2 as well. This indicates that we

are in the most frequent case. In Step 4, if s'_1 is equal to 0, then $\deg(u_c) = 1$. Formulas and operations counts for this special case can be found in [LS05]. Here we give only formulas for the most common case, i.e. $\deg(u_a) = \deg(u_c) = 2$.

Algorithm 3 (DBL22, $h(x) = x^2 + h_1x + h_0$, $f(x) = x^5 + f_1x + f_0$)

INPUT: The divisor class $\overline{D} = [u_a, v_a]$
OUTPUT: The divisor class $[u_c, v_c] = [2]\overline{D}$

1: $z_0 \leftarrow u_{a0}^2$, $z_1 \leftarrow u_{a1}^2$, $w_0 \leftarrow v_{a1}(h_1 + v_{a1})$, $k'_1 = z_1 + v_{a1}$	▷ 1M+2S
2: $w_1 = h_1u_{a0}$, $w_2 = h_0u_{a1}$	▷ 2M
3: $r = h_0^2 + z_0 + (h_1 + u_{a1})(w_1 + w_2)$	▷ 1M
4: $s'_1 = f_1 + z_0 + h_0z_1 + h_1(u_{a1}k'_1 + w_0)$	▷ 3M
5: $m_0 = f_0 + w_1k'_1 + h_0w_0 + v_{a0}^2$, $w_1 = (rs'_1)^{-1}$, $w_2 = rw_1$	▷ 1I+4M+1S
6: $w_3 = s_1'^2w_1$, $w_4 = rw_2$, $w_5 = w_4^2$, $s''_0 = u_{a1} + m_0w_2$	▷ 3M+2S
7: $l'_2 = u_{a1} + s''_0$, $l'_1 = u_{a1}s''_0 + u_{a0}$, $l'_0 = u_{a0}s''_0$	▷ 2M
8: $u_{c0} = s_0''^2 + w_4(s''_0 + u_{a1} + h_1)$, $u_{c1} = w_4 + w_5$	▷ 1M+1S
9: $w_1 = l'_2 + u_{c1}$, $w_2 = u_{c1}w_1 + u_{c0} + l'_1$	▷ 1M
10: $v_{c1} = w_2w_3 + v_{a1} + h_1 + u_{c1}$, $w_2 = u_{c0}w_1 + l'_0$	▷ 2M
11: $v_{c0} = w_2w_3 + v_0 + h_0 + u_{c0}$	▷ 1M
12: **return** $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$	▷ 1I+21M+6S

3.3.3. Type Ib: $h(x) = x^2 + h_1x$

The following algorithm [LS05, Table 2] allows doubling of a divisor class for curves of the form

$$C : y^2 + (x^2 + h_1x)y = x^5 + f_4x^4 + f_1x + f_0 \tag{3.11}$$

over $\mathbb{F}_{2^d}$. If d is odd (which is always the case in this chapter because we choose d to be prime to avoid Weil descent attacks), then we can always obtain $f_4 \in \mathbb{F}_2$. Additionally, if we choose $h_1 = 0$ then the curve equation can be brought to Type Ic.

Assuming $f_4 \in \mathbb{F}_2$ and $h_1 = 0$, the operation count changes from 1I+17M+5S to 1I+10M+6S but we give the algorithm in the more general form.

Algorithm 4 (DBL22, $h(x) = x^2 + h_1x$, $f(x) = x^5 + f_4x^4 + f_1x + f_0$)

INPUT: The divisor class $\overline{D} = [u_a, v_a]$
OUTPUT: The divisor class $[2]\overline{D} = [u_c, v_c]$

1: $z_0 \leftarrow u_{a0}^2$, $z_1 \leftarrow u_{a1}^2$, $w_0 \leftarrow v_{a1}(h_1 + v_{a1})$, $k'_1 \leftarrow z_1 + v_{a1}$ ▷ 1M+2S
2: $z_2 \leftarrow h_1u_{a1}$, $z_3 \leftarrow f_4u_{a1}$, $\tilde{r} \leftarrow u_{a0} + h_1^2 + z_2$ ▷ 2M
3: $w_2 \leftarrow u_{a1}(k'_1 + z_3) + w_0$, $w_3 \leftarrow v_{a0} + h_1k'_1$ ▷ 2M
4: $s'_1 \leftarrow f_1 + z_0 + h_1w_2$, $m_0 \leftarrow w_2 + w_3$ ▷ 1M
5: $w_2 \leftarrow (s'_1)^{-1}$, $w_3 \leftarrow u_{a0}w_2$, $w_4 \leftarrow \tilde{r}w_3$, $w_5 \leftarrow w_4^2$ ▷ 1I+2M+1S
6: $s''_0 \leftarrow u_{a1} + m_0w_3$, $z_4 \leftarrow f_4w_4$, $u_{c1} \leftarrow w_4 + w_5$ ▷ 2M
7: $u_{c0} \leftarrow s''^2_0 + w_4(s''_0 + h_1 + u_{a1} + z_4)$ ▷ 1M+1S
8: $z_5 \leftarrow w_2(m_0^2 + k'_1(s'_1 + h_1m_0))$, $z_6 \leftarrow s''_0 + h_1 + z_4 + z_5$ ▷ 3M+1S
9: $v_{c0} \leftarrow v_{a0} + z_2 + z_1 + w_4(u_{c0} + z_3) + s''_0z_6$ ▷ 2M
10: $v_{c1} \leftarrow v_{a1} + w_4(u_{c1} + s''_0 + f_4 + u_{a1}) + z_5$ ▷ 1M
11: **return** $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$ ▷ 1I+17M+5S
12: ▷ 1I+10M+6S (if $f_4 \in \mathbb{F}_2$ and $h_1 = 0$)

3.3.4. Type Ic: $h(x) = x^2$

In this section, the curve C is of the form

$$C : y^2 + x^2y = x^5 + f_4x^4 + f_1x + f_0, \tag{3.12}$$

where $f_4 \in \mathbb{F}_2$. For this case there are no dedicated doubling or addition formulas published until now. Thus, we refer to Type Ib (i.e. $h(x) = x^2 + h_1x$) in the previous section and use those doubling formulas with $h_1 = 0$. The operation count for the most frequent case DBL22 drops down from 1I+17M+5S to 1I+10M+6S.

3.3.5. Type II: $h(x) = h_1x$

In this case the curve equation is of the form

$$C : y^2 + h_1xy = x^5 + f_3x^3 + f_2x^2 + f_0, \tag{3.13}$$

where $f_2 \in \mathbb{F}_2$. This is the most interesting case because the form of $h(x)$ allows the fastest doubling (and halving) in genus 2. The following algorithm is taken from [LS05, Table 1] but can also be found in [ACD+05, page 339].

Using the isomorphic transformations for Type II in Section 3.1 we can make $h(x)$ monic, i.e. $h_1 = 1$. In this case the operation count of the algorithm decreases noticeably from 1I+9M+5S down to 1I+5M+6S.

Algorithm 5 (DBL22, $h(x) = h_1 x$, $f(x) = x^5 + f_3 x^3 + f_2 x^2 + f_0$, $f_2 \in \mathbb{F}_2$)

INPUT: The divisor class $\overline{D} = [u_a, v_a]$, h_1^2 and h^{-1}
OUTPUT: The divisor class $[u_c, v_c] = [2]\overline{D}$

1: $z_0 \leftarrow u_{a0}^2$, $k_1' \leftarrow u_{a1}^2 + f_3$, $w_0 \leftarrow f_0 + v_{a0}^2$, $w_1 \leftarrow w_0^{-1} z_0$ ▷ 1I+1M+3S
2: $z_1 \leftarrow k_1' w_1$, $s_0'' \leftarrow z_1 + u_{a1}$, $w_2 \leftarrow h_1^2 w_1$ ▷ 2M
3: $u_{c1} \leftarrow w_2 w_1$, $u_{c0} \leftarrow s_0''^2 + w_2$, $w_3 \leftarrow w_2 + k_1'$ ▷ 1M+1S
4: $v_{c1} = h_1^{-1}(w_3 z_1 + w_2 u_{c1} + f_2 + v_{a1}^2)$ ▷ 3M+1S
5: $v_{c0} = h_1^{-1}(w_3 u_{c0} + f_1 + z_0)$ ▷ 2M
6: **return** $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$ ▷ 1I+9M+5S (1I+5M+6S if $h_1 = 1$)

Note that in Step 1, if $w_0 = 0$ we are in the DBL21 case, i.e. the polynomial u_c of the doubled divisor class has degree 1. As in the previous cases we assume to be solely in the most frequent case DBL22. Formulas for the special case DBL21 can be found in [LS05, Table 1] and [ACD+05, page 339]. For inversion-free formulas we refer to the doubling algorithms in Section 3.5.

3.4. Halving

In this section we give explicit halving formulas for each type of the classification from Section 3.1.2, except for Type Ib. The 2-torsion subgroup of the Picard group of a Type-Ib curve has order 4, and the computation of a preimage of the doubling in the odd-order subgroup is therefore significantly more costly than the appropriate doubling. Therefore, we exclude this case from our considerations.

3.4.1. Type II: $h(x) = x$

In this section, the curve C is of the form

$$C : y^2 + xy = x^5 + f_3 x^3 + f_2 x^2 + f_0, \tag{3.14}$$

where $f_2 \in \mathbb{F}_2$.

Theorem 3.4. Let $\overline{D}_a = [u_a, v_a]$ be a divisor class in $\mathrm{Pic}^0(C)$. If $\deg(u_a) = 2$, then $\overline{D}_a$ can be halved if and only if $\mathrm{TR}(u_{a1}(u_{a0} + f_3 + u_{a1}^2)) = 0$. If $\deg(u_a) = 1$, then $\overline{D}_a$ can be halved if and only if $\mathrm{TR}(f_2 + u_{a0}(u_{a0}^2 + f_3)) = 0$.

Proof. We consider the doubling as described in Cantor's algorithm (Algorithm 1) and reverse its steps to investigate the halving of $\overline{D}_a$. One of the steps that we will take back is computing $\mathbb{F}_{2^d}$-rational solutions to a quadratic equation of the form $x^2+x+c=0$. This equation has solutions in $\mathbb{F}_{2^d}$ if and only if $\mathrm{TR}(c)=0$ (cf. Section 11.2.6 in [ACD+05]), i.e. $\overline{D}_a$ can be halved if and only if $\mathrm{TR}(c)=0$. We give the proof for the case $\deg(u_a)=2$ and thus show that $c=u_{a1}(u_{a0}+f_3+u_{a1}^2)$, as stated in the theorem. The case $\deg(u_a)=1$ is done analogously.

In Cantor's algorithm, Step 4 (combination step) computes a polynomial $u=x^4+u_2x^2+u_0$ by squaring the degree-2 polynomial $u_1=x^2+u_{11}x+u_{10}$ from the input divisor class. (For simplicity, we use the notation of Algorithm 1 here and adapt it to our notation at the end of the proof.) In the reduction (Steps 5 to 8), the degree of u will be reduced to 1 or 2. Since we consider the case HLV22, the reduced polynomial u' has degree 2 and is equal to $x^2+u_1'x+u_0'$. Since v and v' in Cantor's algorithm are computed modulo u and u', the degree of v is less than 4 and the degree of v' is less than 2. We obtain $v=v_3x^3+v_2x^2+v_1x+v_0$, $v'=v_1'x+v_0'$ and get the following relations from the reduction step by equating the coefficients of the polynomials:

(1) $v_3^2u_1'=1$,

(2) $v_3^2u_0'=v_3+v_2^2+v_3^2+v_3^2u_2$,

(3) $0=v_2+f_3+u_2$.

From (1), we get $v_3=1/\sqrt{u_1'}$. Writing (2) with $u_2=v_2+f_3$ from (3), we get a relation to compute v_2:

$$v_3^2u_0'=v_3+v_2^2+v_3^2(v_2+f_3). \tag{3.15}$$

This can be written as the quadratic equation

$$v_2^2+v_3^2v_2+(v_3^2f_3+v_3+v_3^2u_0') \tag{3.16}$$

in the variable v_2. Dividing the equation by v_3^4, writing $c=(v_3^2f_3+v_3+v_3^2u_0')/v_3^4$ and applying the transformation $x\leftarrow v_2/v_3^2$ (this procedure is explained in Section 11.2.6 in [ACD+05]) gives:

$$v_2=v_3^2\cdot\mathrm{RootOf}\left(x^2+x+(v_3+v_3^2(f_3+u_0'))/v_3^4=0\right). \tag{3.17}$$

The factor v_3^2 at the beginning of (3.17) corrects the transformation $x\leftarrow v_2/v_3^2$. Using relation (1), we can write this as

$$v_2=v_3^2\cdot\mathrm{RootOf}\left(x^2+x+u_1'(\sqrt{u_1'}+f_3+u_0')=0\right). \tag{3.18}$$

A solution for v_2 exists if and only if $\mathrm{TR}(u_1'(\sqrt{u_1'}+f_3+u_0'))=0$, i.e. the divisor class can be halved if and only if this trace is equal to 0. Using the fact that the

trace is additive and that $\mathrm{TR}(a) = \mathrm{TR}(a^2)$ (see (1) and (3) in Lemma 3.3), we can write

$$\mathrm{TR}(u_1'(\sqrt{u_1'} + f_3 + u_0')) = \mathrm{TR}(u_1'(u_1'^2 + f_3 + u_0')). \tag{3.19}$$

With this, we replace 1SR by 1S, which gives a small performance increase in the computation because 1S can be faster computed than 1SR. Using our notation, we see that $\overline{D}_a$ can be halved if and only if $\mathrm{TR}(u_{a1}(u_{a1}^2 + f_3 + u_{a0})) = 0$, as claimed in the theorem. □

From Theorem 3.4, we obtain a simple condition for the group order:

Corollary 3.5. The Picard group of the curve C given by (3.14) has order $2r$, where r odd, if and only if $f_2 = 1$.

Proof. The Picard group of the curve C has exactly one divisor class of order 2, namely $[x, \sqrt{f_0}]$. The group order is divisible by 4 if and only if $[x, \sqrt{f_0}]$ can be halved. From Theorem 3.4, this is possible if and only if $\mathrm{TR}(f_2) = 0$. Since $f_2 \in \mathbb{F}_2$ we find that $\mathrm{Pic}^0(C)$ has a divisor class of order 4 if and only if $f_2 = 0$. □

Observe that if $\overline{D}_{c_1}$ and $\overline{D}_{c_2}$ are the two preimages of $\overline{D}_a$ under the doubling, then $\overline{D}_{c_1} - \overline{D}_{c_2} = [x, \sqrt{f_0}] = \overline{D}_{c_2} - \overline{D}_{c_1}$, i.e. the difference of two preimages is the unique divisor class of order 2.

Lemma 3.6. Let $\overline{D}_a = [u_a, v_a]$ be a divisor class in $\mathrm{Pic}^0(C)$ that can be halved and let $\overline{D}_c = [u_c, v_c] = [\frac{1}{2}]\overline{D}_a$ be its preimage (under the doubling) of odd order. Then the following holds:

(1) If $\deg(u_a) = 1$, then $\deg(u_c) = 2$, and we are in case HLV12.

(2) If $\deg(u_a) = 2$ and $u_{a1} \neq 0$, then $\deg(u_c) = 2$, and we are in case HLV22.

(3a) If $\deg(u_a) = 2$, $u_{a1} = 0$ and $\mathrm{TR}(f_2 + \sqrt{u_{a0}}(u_{a0} + f_3)) = 0$, then $u_c = x + \sqrt{u_{a0}}$, $v_c = v_{a0} + \sqrt{u_{a0}}v_{a1}$ and we are in case HLV21.

(3b) If $\deg(u_a) = 2$, $u_{a1} = 0$ and $\mathrm{TR}(f_2 + \sqrt{u_{a0}}(u_{a0} + f_3)) = 1$, then $u_c = x^2 + \sqrt{u_{a0}}x$, and we are in case HLV22.

Proof. In (1), the degree of u_a is 1. According to Section 3.3.1, the only case with $\deg(u_a) = 1$ is HLV12.

In (2), the degree of u_a is 2 and $u_{a1} \neq 0$. Let us assume $\deg(u_c) = 1$, i.e. $u_c = x + u_{c0}$. Applying Cantor's algorithm to double $[u_c, v_c]$ gives $u_a = x^2 + u_{c0}^2$ with $u_{a1} = 0$, which is a contradiction. Hence $\deg(u_c) = 2$, and we are in the case HLV22.

In (3a) and (3b), we have $\deg(u_a) = 2$ and $u_{a1} = 0$, i.e. $u_a = x^2 + u_{a0}$. A possible choice for u_c is $\sqrt{u_a} = x + \sqrt{u_{a0}}$, because this takes Step 4 of Algorithm 1 back. Now we have two possibilities for the preimage of $[u_a, v_a]$, namely

$$[x + \sqrt{u_{a0}}, v_c] \quad \text{or} \quad [x + \sqrt{u_{a0}}, v_c] \oplus [x, \sqrt{f_0}].$$

The second preimage results from adding the unique divisor class of order 2 to the first one. The polynomial u_c of the second preimage is equal to $x^2+\sqrt{u_{a0}}x$. With Theorem 3.4 we can now check which of the two divisor classes can be halved to see which one has odd order. The check can be done using the trace conditions in Theorem 3.4. If the first divisor class has odd order (i.e. $\mathrm{TR}(f_2+\sqrt{u_{a0}}(u_{a0}+f_3))=0$), then we are in the case HLV21 and the preimage is $[x+\sqrt{u_{a0}}, v_{a0}+\sqrt{u_{a0}}v_{a1}]$, which can be verified by Step 4 of Cantor's algorithm. Otherwise, the second divisor class has odd order and $\mathrm{TR}(f_2+\sqrt{u_{a0}}(u_{a0}+f_3))=1$. Thus $u_c=x^2+\sqrt{u_{a0}}x$, and we are in the case HLV22. □

In Cases (1) and (2), we still need to decide which of the two possible divisor classes is the halved class in the subgroup of odd order. We use Theorem 3.4 on u_c to ensure that the corresponding divisor class can be halved again and we correct the computations if necessary. We obtain the following formulas which have been verified with the help of Magma [BCP97]:

Algorithm 6 (HLV22, $h(x)=x$, $f(x)=x^5+f_3x^3+x^2+f_0$)

INPUT: The divisor class $\overline{D}_a=[x^2+u_{a1}x+u_{a0}, v_{a1}x+v_{a0}]$

OUTPUT: The divisor class $[x^2+u_{c1}x+u_{c0}, v_{c1}x+v_{c0}]=[\frac{1}{2}]\overline{D}_a$

1: $s_0 \leftarrow \sqrt{u_{a1}}$, $s_1 \leftarrow s_0^{-1}$, $s_2 \leftarrow s_1^2$ ▷ 1I+1S+1SR

2: $s_3 \leftarrow \mathrm{HT}(u_{a1}(u_{a0}+f_3+s_0))$, $s_4 \leftarrow s_3s_2$, $s_5 \leftarrow s_4+f_3$ ▷ 2M+1HT

3: $s_6 \leftarrow v_{a0}+u_{a0}(s_4+s_0)$, $s_7 \leftarrow s_6$, $u_{c1} \leftarrow \sqrt{s_5}$ ▷ 1M+1SR

4: **if** $\mathrm{TR}(s_5(s_7+f_3^2+u_{c1}))=1$ **then** ▷ 1M+1TR

5: $\quad s_3 \leftarrow s_3+1$, $s_4 \leftarrow s_4+s_2$, $s_5 \leftarrow s_5+s_2$

6: $\quad s_6 \leftarrow s_6+s_2u_{a0}$, $s_7 \leftarrow s_6$, $u_{c1} \leftarrow u_{c1}+s_1$ ▷ 1M

7: **end if**

8: $u_{c0} \leftarrow \sqrt{s_7}$, $v_{c0} \leftarrow \sqrt{s_7s_1+f_0}$ ▷ 1M+2SR

9: $v_{c1} \leftarrow v_{a1}+1+s_3+s_1(u_{a0}+u_{a1}^2+u_{c0}+s_5)+s_4u_{c1}$ ▷ 2M+1S

10: **return** $[x^2+u_{c1}x+u_{c0}, v_{c1}x+v_{c0}]$

11: ▷ Worst case: 1I+8M+2S+4SR+1HT+1TR

12: ▷ Average: 1I+7.5M+2S+4SR+1HT+1TR

Algorithm 7 (HLV12, $h(x) = x$, $f(x) = x^5 + f_3x^3 + x^2 + f_0$)

INPUT: The divisor class $\overline{D}_a = [x + u_{a0}, v_{a0}]$
OUTPUT: The divisor class $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}] = [\frac{1}{2}]\overline{D}_a$

1: $s_0 \leftarrow \sqrt{u_{a0}}$, $s_1 \leftarrow f_3 + s_0$, $s_2 \leftarrow 1 + s_1u_{a0}$ ▷ 1M+1SR
2: $s_3 \leftarrow \mathrm{HT}(s_2)$, $s_4 \leftarrow v_{a0} + u_{a0}(s_3 + 1 + s_0u_{a0})$ ▷ 2M+1HT
3: $u_{c1} \leftarrow \sqrt{s_1}$, $u_{c0} \leftarrow \sqrt{s_4}$ ▷ 2SR
4: **if** $\mathrm{TR}(u_{c1}(u_{c0} + s_1 + f_3)) = 1$ **then** ▷ 1M+1TR
5: $s_3 \leftarrow s_3 + 1$, $s_4 \leftarrow s_4 + u_{a0}$, $u_{c0} \leftarrow \sqrt{s_4}$ ▷ 1SR
6: **end if**
7: $v_{c1} \leftarrow s_3 + s_0u_{c1}$, $v_{c0} \leftarrow s_4 + s_0u_{c0}$ ▷ 2M
8: **return** $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$ ▷ Worst case: 6M+4SR+1TR+1HT
9: ▷ Average: 6M+3.5SR+1TR+1HT

Algorithm 8 (HLV21/22, $h(x) = x$, $f(x) = x^5 + f_3x^3 + x^2 + f_0$)

INPUT: The divisor class $\overline{D}_a = [x^2 + u_{a0}, v_{a1}x + v_{a0}]$
OUTPUT: The divisor class $[x + u_{c0}, v_{c0}] = [\frac{1}{2}]\overline{D}_a$
or $[x^2 + u_{c1}x, v_{c1}x + v_{c0}] = [\frac{1}{2}]\overline{D}_a$

1: $s_0 \leftarrow \sqrt{u_{a0}}$, $s_1 \leftarrow v_{a0} + s_0v_{a1}$ ▷ 1M+1SR
2: **if** $\mathrm{TR}(u_{a0}s_0) = 1$ **then** ▷ 1M+1TR
3: $\overline{E} \leftarrow [x + s_0, s_1]$
4: **else**
5: $s_2 \leftarrow s_0^{-1}$, $v_{c1} \leftarrow (s_1 + \sqrt{f_0})s_2$ ▷ 1I+1M
6: $v_{c0} \leftarrow s_1 + v_{c1}s_0$, $\overline{E} \leftarrow [x^2 + s_0x, v_{c1}x + v_{c0}]$ ▷ 1M
7: **end if**
8: **return** $\overline{E}$ ▷ Worst case: 1I+4M+1SR+1TR
9: ▷ Average: 0.5I+3M+1SR+1TR

3.4.2. Type Ia: $h(x) = x^2 + x + 1$

In this section, the curve C is of the form

$$C : y^2 + (x^2 + x + 1)y = f_5x^5 + f_4x^4 + f_1x + f_0, \tag{3.20}$$

where $f_4 \in \mathbb{F}_2$ and $f_4 \cdot \mathrm{TR}(f_5) = 0$. To improve the efficiency of the halving formulas, we will assume that f_5^{-1} and f_5^{-2} are precomputed. Note that this computation is necessary only once per curve and the values are published along with the system parameters.

Theorem 3.7. Let $\overline{D} = [u_a, v_a]$ be a divisor class in $\mathrm{Pic}^0(C)$. If $\deg(u_a) = 2$, then $\overline{D}$ can be halved if and only if $\mathrm{TR}(u_{a1} f_5) = 0$. If $\deg(u_a) = 1$, then $\overline{D}$ can be halved if and only if $\mathrm{TR}(f_4 + f_5 u_{a0}) = 0$.

Proof. To prove this, we can follow the same ideas as in Theorem 3.4, with one difference: some of the halving formulas require to solve two quadratic equations rather than one. In those formulas, it is easy to verify that switching between the roots of the first quadratic equation (adding 1 to the half-trace) changes the trace of the constant term of the second quadratic equation by 1. The choice of root for the first equation (when roots exist in $\mathbb{F}_{2^d}$) and the trace condition from the second equation are then purely internal to the halving formula. □

From this theorem, we obtain a simple condition for the group order:

Corollary 3.8. The Picard group of the curve C given by (3.20) has order $2r$, where r is odd, if and only if $\mathrm{TR}(f_5) = 1$ and $f_4 = 0$.

Proof. The Picard group of the curve C has exactly one divisor class of order 2, which is of the form $[x^2 + x + 1, v_h]$. The order of the Picard group is divisible by 4 if and only if $[x^2 + x + 1, v_h]$ can be halved. From Theorem 3.7, this is possible if and only if $\mathrm{TR}(f_5) = 0$. The condition $f_4 = 0$ follows directly from the restriction $f_4 \cdot \mathrm{TR}(f_5) = 0$. □

Observe that if $\overline{D}_{c_1}$ and $\overline{D}_{c_2}$ are the two preimages of $\overline{D}_a$ under the doubling, then $\overline{D}_{c_1} - \overline{D}_{c_2} = [x^2 + x + 1, v_h] = \overline{D}_{c_2} - \overline{D}_{c_1}$, i.e. the difference of two preimages is the unique divisor class of order 2.

Lemma 3.9. Let $\overline{D}_a = [u_a, v_a]$ be a divisor class in $\mathrm{Pic}^0(C)$ that can be halved and let $\overline{D}_c = [u_c, v_c] = [\frac{1}{2}]\overline{D}_a$ be its preimage (under the doubling) of odd order. Then the following holds:

(1) If $\deg(u_a) = 1$, then $\deg(u_c) = 2$, and we are in case HLV12.

(2) If $\deg(u_a) = 2$ and $u_{a1} \neq 0$, then $\deg(u_c) = 2$, and we are in case HLV22.

(3a) If $\deg(u_a) = 2$, $u_{a1} = 0$ and $\mathrm{TR}(f_4 + f_5\sqrt{u_{a0}}) = 0$, then we are in case HLV21.

(3b) If $\deg(u_a) = 2$, $u_{a1} = 0$ and $\mathrm{TR}(f_4 + f_5\sqrt{u_{a0}}) = 1$, then we are in case HLV22.

Proof. In (1), the degree of u_a is 1. According to Section 3.3.1, the only case with $\deg(u_a) = 1$ is HLV12.

In (2), the degree of u_a is 2 and $u_{a1} \neq 0$. Let us assume $\deg(u_c) = 1$, i.e. $u_c = x + u_{c0}$. Applying Cantor's algorithm to double $[u_c, v_c]$ gives $u_a = x^2 + u_{c0}^2$ with $u_{a1} = 0$, which is a contradiction. Hence $\deg(u_c) = 2$, and we are in the case HLV22.

In (3a) and (3b), we have $\deg(u_a) = 2$ and $u_{a1} = 0$, i.e. $u_a = x^2 + u_{a0}$. A possible choice for u_c is $\sqrt{u_a} = x + \sqrt{u_{a0}}$, because this takes Step 4 of Algorithm 1 back. Now we have two possibilities for the preimage of $[u_a, v_a]$, namely

$$[x + \sqrt{u_{a0}}, v_c] \quad \text{or} \quad [x + \sqrt{u_{a0}}, v_c] \oplus [x^2 + x + 1, v_h].$$

The second preimage results from adding the unique divisor class of order 2 to the first divisor class. With Theorem 3.7 we can now check which of the two divisor classes can be halved to see which one has odd order. The check can be done using the trace conditions in Theorem 3.7. If the first divisor class has odd order (i.e. $\mathrm{TR}(f_4 + f_5\sqrt{u_{a0}}) = 0$), then we are in the case HLV21 and $u_c = x + \sqrt{u_{a0}}$. Otherwise, the second divisor class has odd order, and we are in the case HLV22. □

In Cases (1) and (2), we still need to decide which of the two possible divisor classes is the halved class in the subgroup of odd order. We use Theorem 3.7 on u_c to ensure that the corresponding divisor class can be halved again and we correct the computations if necessary. We obtain the following formulas which have been verified with the help of Magma [BCP97]:

Algorithm 9 (HLV22, $h(x) = x^2 + x + 1$, $f(x) = f_5x^5 + f_1x + f_0$)

INPUT: The divisor class $\overline{D} = [x^2 + u_{a1}x + u_{a0}, v_{a1}x + v_{a0}]$, and f_5^{-2} precomputed

OUTPUT: The divisor class $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}] = [\frac{1}{2}]\overline{D}$

1: $s_0 \leftarrow f_5u_{a1}$, $s_1 \leftarrow u_{a1}^{-1}$, $s_2 \leftarrow \mathrm{HT}(s_0)$, $s_3 \leftarrow s_2s_1$ ▷ 1I+2M+1HT

2: $s_4 \leftarrow s_2u_{a1}$, $s_5 \leftarrow f_5u_{a0} + s_2 + s_3 + v_{a1} + 1 + u_{a1} + s_4$ ▷ 2M

3: $s_6 \leftarrow s_5u_{a1}$ ▷ 1M

4: **if** $\mathrm{TR}(s_6) = 1$ **then** ▷ 1TR

5: $\quad s_2 \leftarrow s_2 + 1$, $s_3 \leftarrow s_3 + s_1$, $s_4 \leftarrow s_4 + u_{a1}$

6: $\quad s_5 \leftarrow s_5 + 1 + s_1 + u_{a1}$, $s_6 \leftarrow s_5u_{a1}$ ▷ 1M

7: **end if**

8: $s_8 \leftarrow \mathrm{HT}(s_6)$, $s_9 \leftarrow s_8s_1$ ▷ 1M+1HT

9: $s_{10} \leftarrow s_3u_{a0} + s_4 + s_8 + v_{a1} + 1 + u_{a1}$ ▷ 1M

10: **if** $\mathrm{TR}((s_{10} + s_3 + s_9)(s_3 + f_5 + s_1)) = 1$ **then** ▷ 1M+1TR
11: $\quad s_8 \leftarrow s_8 + 1,\ s_9 \leftarrow s_9 + s_1,\ s_{10} \leftarrow s_{10} + 1$
12: **end if**
13: $s_{11} \leftarrow (s_3 + f_5 + s_1)f_5^{-2},\ s_{12} \leftarrow (s_{10} + s_3 + s_9)s_{11}$ ▷ 2M
14: $s_{13} \leftarrow (s_2 + s_9)u_{a0} + v_{a0} + 1 + u_{a0}$ ▷ 1M
15: $s_{14} \leftarrow (s_{13} + s_{10} + f_1)s_{11},\ u_{c0} \leftarrow \sqrt{s_{14}},\ u_{c1} \leftarrow \sqrt{s_{12}}$ ▷ 1M+2SR
16: $s_{15} \leftarrow s_3 u_{c1},\ s_{16} \leftarrow s_{15} + s_9,\ s_{17} \leftarrow s_{16}u_{c0}$ ▷ 2M
17: $v_{c1} \leftarrow s_{10} + (s_3 + s_{16})(u_{c0} + u_{c1}) + s_{15} + s_{17}$ ▷ 1M
18: $v_{c0} \leftarrow s_{13} + s_{17}$
19: **return** $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$
20: ▷ Worst case: 1I+16M+2SR+2TR+2HT
21: ▷ Average: 1I+15.5M+2SR+2TR+2HT

Note that for curves with $f_5 = 1$, the worst-case operation count decreases to 1I+13M+2SR+2HT+2TR. If we move the computation of s_{11} and s_{12} before the second trace computation (which becomes $\mathrm{TR}(s_{12})$), then the average operation count drops to 1I+12M+2SR+2HT+2TR (with that approach, a multiplication is required to correct s_{12}).

Algorithm 10 (HLV12, $h(x) = x^2 + x + 1$, $f(x) = f_5x^5 + f_1x + f_0$)

INPUT: The divisor class $\overline{D}_a = [x + u_{a0}, v_{a0}]$ and f_5^{-1} precomputed
OUTPUT: The divisor class $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}] = [\frac{1}{2}]\overline{D}_a$

1: $s_0 \leftarrow \mathrm{HT}(f_5u_{a0}),\ s_1 \leftarrow u_{a0}^2 + u_{a0}$ ▷ 1M+1S+1HT
2: $s_2 \leftarrow v_{a0} + 1 + s_1(s_0 + 1) + s_0^2$ ▷ 1M+1S
3: **if** $\mathrm{TR}(s_2) = 1$ **then** ▷ 1TR
4: $\quad s_0 \leftarrow s_0 + 1,\ s_2 \leftarrow s_2 + s_1 + 1$
5: **end if**
6: $s_3 \leftarrow \mathrm{HT}(s_2)$ ▷ 1HT
7: **if** $\mathrm{TR}(s_3f_5) = 1$ **then** $s_3 \leftarrow s_3 + 1$ **end if** ▷ 1M+1TR
8: $s_4 \leftarrow s_3f_5^{-1},\ s_5 \leftarrow s_0 + s_3,\ s_6 \leftarrow s_0^2 + s_3(1 + u_{a0} + s_3)$ ▷ 2M+1S
9: $s_7 \leftarrow (f_1 + s_5 + s_6)f_5^{-1},\ u_{c1} \leftarrow \sqrt{s_4},\ u_{c0} \leftarrow \sqrt{s_7}$ ▷ 1M+2SR
10: $v_{c1} \leftarrow s_5 + s_0u_{c1},\ v_{c0} \leftarrow s_6 + s_0u_{c0}$ ▷ 2M
11: **return** $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$ ▷ 8M+3S+2SR+2TR+2HT

Algorithm 11 (HLV21/22, $h(x) = x^2 + x + 1$, $f(x) = f_5x^5 + f_1x + f_0$)

INPUT: The divisor class $\overline{D}_a = [x^2 + u_{a0}, v_{a1}x + v_{a0}]$ and f_5^{-2}
OUTPUT: The divisor class $[x + u_{c0}, v_{c0}] = [\frac{1}{2}]\overline{D}_a$
or $[x^2 + u_{c1}x, v_{c1}x + v_{c0}] = [\frac{1}{2}]\overline{D}_a$

1: $u_{c0} \leftarrow \sqrt{u_{a0}}$ ▷ 1SR
2: **if** $\mathrm{TR}(f_5u_{c0}) = 0$ **then** ▷ 1M+1TR
3: $v_{c0} \leftarrow v_{a0} + u_{c0}v_{a1}$ ▷ 1M
4: **return** $[x + u_{c0}, v_{c0}]$
5: **else**
6: $s_0 \leftarrow v_{a1} + 1 + u_{a0}f_5$, $s_1 \leftarrow s_0 + f_1$, $s_2 \leftarrow s_0 + f_5$ ▷ 1M
7: $s_3 \leftarrow u_{a0} + (s_0 + s_2^2)f_5^{-2}$, $u_{c1} \leftarrow \sqrt{s_3}$, $s_4 \leftarrow f_5u_{c1}$, ▷ 2M+1SR+1S
8: $s_5 \leftarrow s_3u_{a0} + (s_2 + s_0^2 + f_1)f_5^{-2}$, $u_{c0} \leftarrow \sqrt{s_5}$ ▷ 2M+1SR+1S
9: $s_6 \leftarrow (s_2 + s_4)u_{c0}$, $v_{c1} \leftarrow (s_0 + s_4)(u_{c1} + u_{c0} + 1) + s_6$ ▷ 2M
10: $v_{c0} \leftarrow s_1 + s_6$
11: **return** $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$
12: **end if**
13: ▷ Worst case: 8M+3SR+2S+1TR
14: ▷ Average: 5M+2SR+1S+1TR

3.4.3. Type Ic: $h(x) = x^2$

In this section, the curve C is of the form

$$C : y^2 + x^2y = x^5 + f_4x^4 + f_1x + f_0, \tag{3.21}$$

where $f_4 \in \mathbb{F}_2$.

Theorem 3.10. Let $\overline{D} = [u_a, v_a]$ be a divisor class in $\mathrm{Pic}^0(C)$. If $\deg(u_a) = 2$, then $\overline{D}$ can be halved if and only if $\mathrm{TR}(u_{a1}) = 0$. If $\deg(u_a) = 1$, then $\overline{D}$ can be halved if and only if $\mathrm{TR}(f_4 + u_{a0}) = 0$.

Proof. As in Theorem 3.4. □

From this theorem, we obtain a simple condition for the group order:

Corollary 3.11. The Picard group of the curve C given by (3.21) has order $2r$, where r is odd, if and only if $f_4 = 1$.

Proof. The Picard group of the curve C has exactly one divisor class of order 2, which is of the form $[x, \sqrt{f_0}]$. The order of the Picard group is divisible by 4 if and only if $[x, \sqrt{f_0}]$ can be halved. From Theorem 3.10, this is possible if and only if $\mathrm{TR}(f_4) = 0$. Since $f_4 \in \mathbb{F}_2$, the Picard group of C has a divisor class of order 4 if and only if $f_4 = 0$. □

Observe that if $\overline{D}_{c_1}$ and $\overline{D}_{c_2}$ are the two preimages of $\overline{D}_a$ under the doubling, then $\overline{D}_{c_1} - \overline{D}_{c_2} = [x, \sqrt{f_0}] = \overline{D}_{c_2} - \overline{D}_{c_1}$, i.e. the difference of two preimages is the unique divisor class of order 2.

Lemma 3.12. Let $\overline{D}_a = [u_a, v_a]$ be a divisor class in $\mathrm{Pic}^0(C)$ that can be halved and let $\overline{D}_c = [u_c, v_c] = [\frac{1}{2}]\overline{D}_a$ be its preimage (under the doubling) of odd order. Then the following holds:

(1) If $\deg(u_a) = 1$, then $\deg(u_c) = 2$, and we are in case HLV12.

(2) If $\deg(u_a) = 2$ and $u_{a1} \neq 0$, then $\deg(u_c) = 2$, and we are in case HLV22.

(3a) If $\deg(u_a) = 2$, $u_{a1} = 0$ and $\mathrm{TR}(f_4 + u_{a0}) = 0$, then $u_c = x + \sqrt{u_{a0}}$, $v_c = v_{a0} + \sqrt{u_{a0}} v_{a1}$ and we are in case HLV21.

(3b) If $\deg(u_a) = 2$, $u_{a1} = 0$ and $\mathrm{TR}(\sqrt{u_{a0}}) = 0$, then $u_c = x^2 + \sqrt{u_{a0}}x$, and we are in case HLV22.

Proof. In (1), the degree of u_a is 1. According to Section 3.3.1, the only case with $\deg(u_a) = 1$ is HLV12.

In (2), the degree of u_a is 2 and $a_{a1} \neq 0$. Let us assume $\deg(u_c) = 1$, i.e. $u_c = x + u_{c0}$. Applying Cantor's algorithm to double $[u_c, v_c]$ gives $u_a = x^2 + u_{c0}^2$ with $u_{a1} = 0$, which is a contradiction. Hence $\deg(u_c) = 2$, and we are in the case HLV22.

In (3a) and (3b), we have $\deg(u_a) = 2$ and $u_{a1} = 0$, i.e. $u_a = x^2 + u_{a0}$. A possible choice for u_c is $\sqrt{u_a} = x + \sqrt{u_{a0}}$, because this takes Step 4 of Algorithm 1 back. Now we have two possibilities for the preimage of $[u_a, v_a]$, namely

$$[x + \sqrt{u_{a0}}, v_c] \quad \text{or} \quad [x + \sqrt{u_{a0}}, v_c] \oplus [x, \sqrt{f_0}].$$

The second preimage results from adding the unique divisor class of order 2 to the first one. The polynomial u_c of the second preimage is equal to $x^2 + \sqrt{u_{a0}}x$. With Theorem 3.10 we can now check which of the two divisor classes can be halved to see which one has odd order. The check can be done using the trace conditions in Theorem 3.10. If the first divisor class has odd order (i.e. $\mathrm{TR}(f_4 + u_{a0}) = 0$), then we are in the case HLV21 and the preimage is $[x + \sqrt{u_{a0}}, v_{a0} + \sqrt{u_{a0}} v_{a1}]$, which can be verified by Step 4 of Cantor's algorithm. Otherwise, the second divisor class has odd order and $\mathrm{TR}(\sqrt{u_{a0}}) = 0$. Thus $u_c = x^2 + \sqrt{u_{a0}}x$, and we are in the case HLV22. □

In Cases (1) and (2), we still need to decide which of the two possible divisor classes is the halved class in the subgroup of odd order. We use Theorem 3.10 on u_c to ensure that the corresponding divisor class can be halved again and we correct the computations if necessary. We obtain the following formulas which have been verified with the help of Magma [BCP97]:

Algorithm 12 (HLV22, $h(x) = x^2$, $f(x) = x^5 + x^4 + f_1x + f_0$)

INPUT: The divisor class $\overline{D} = [x^2 + u_{a1}x + u_{a0}, v_{a1}x + v_{a0}]$
OUTPUT: The divisor class $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}] = [\frac{1}{2}]\overline{D}$

1: $s_0 \leftarrow u_{a1}^{-1}$, $s_1 \leftarrow \mathrm{HT}(u_{a1})$, $s_2 \leftarrow s_1s_0$ ▷ 1I+1M+1HT
2: $s_3 \leftarrow f_1(s_2 + 1 + s_0)$, $s_4 \leftarrow \sqrt{s_1 + (v_{a1} + u_{a0})s_0}$ ▷ 2M+1SR
3: $s_5 \leftarrow v_{a1} + u_{a1}(1 + s_4 + s_1) + s_2u_{a0}$, $s_6 \leftarrow s_5(s_2 + 1 + s_0)$ ▷ 3M
4: **if** $\mathrm{TR}(s_6) = 1$ **then** ▷ 1TR
5: $s_2 \leftarrow s_2 + s_0$, $s_3 \leftarrow s_3 + f_1s_0$, $s_4 \leftarrow s_4 + 1$ ▷ 1M
6: $s_5 \leftarrow s_5 + u_{a0}s_0$, $s_6 \leftarrow s_5(s_2 + 1 + s_0)$ ▷ 2M
7: **end if**
8: $u_{c1} \leftarrow \sqrt{s_6}$, $u_{c0} \leftarrow \sqrt{s_3}$, $v_{c0} \leftarrow \sqrt{f_0 + s_3(1 + s_4)}$ ▷ 1M+3SR
9: $v_{c1} \leftarrow s_6 + s_2u_{c0} + s_4u_{c1}$ ▷ 2M
10: **return** $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$
11: ▷ Worst case: 1I+12M+4SR+1HT+1TR
12: ▷ Average: 1I+10.5M+4SR+1HT+1TR

Algorithm 13 (HLV21/22, $h(x) = x^2$, $f(x) = x^5 + f_4x^4 + f_1x + f_0$)

INPUT: The divisor class $\overline{D}_a = [x^2 + u_{a0}, v_{a1}x + v_{a0}]$
OUTPUT: The divisor class $[x + u_{c0}, v_{c0}] = [\frac{1}{2}]\overline{D}_a$
or $[x^2 + u_{c1}x, v_{c1}x + v_{c0}] = [\frac{1}{2}]\overline{D}_a$

1: $s_0 \leftarrow \sqrt{u_{a0}}$, $s_1 \leftarrow v_{a0} + s_0v_{a1}$ ▷ 1M+1SR
2: **if** $\mathrm{TR}(s_0) = 1$ **then** ▷ 1TR
3: $\overline{E} \leftarrow [x + s_0, s_1]$
4: **else**
5: $s_2 \leftarrow s_0^{-1}$, $v_{c1} \leftarrow (s_1 + \sqrt{f_0})s_2$ ▷ 1I+1M
6: $v_{c0} \leftarrow s_1 + v_{c1}s_0$, $\overline{E} \leftarrow [x^2 + s_0x, v_{c1}x + v_{c0}]$ ▷ 1M
7: **end if**

8: **return** $\overline{E}$ ▷ Worst case: 1I+3M+1SR+1TR
9: ▷ Average: 0.5I+2M+1SR+1TR

Algorithm 14 (HLV12, $h(x) = x^2$, $f(x) = x^5 + f_4x^4 + f_1x + f_0$)

INPUT: The divisor class $\overline{D}_a = [x + u_{a0}, v_{a0}]$
OUTPUT: The divisor class $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}] = [\frac{1}{2}]\overline{D}_a$

1: $s_1 \leftarrow \mathrm{HT}(u_{a0} + 1)$, $s_2 \leftarrow \sqrt{v_{a0} + (s_1 + 1)u_{a0}^2}$ ▷ 1M+1S+1SR+1HT
2: $s_3 \leftarrow s_2(u_{a0} + s_2)$, $s_4 \leftarrow s_2$ ▷ 1M
3: **if** $\mathrm{TR}(s_4) = 1$ **then** ▷ 1TR
4: $s_1 \leftarrow s_1 + 1$, $s_2 \leftarrow s_2 + u_{a0}$, $s_4 \leftarrow s_2$
5: **end if**
6: $u_{c1} \leftarrow \sqrt{s_4}$, $u_{c0} \leftarrow \sqrt{f_1}$, $v_{c1} \leftarrow s_2 + s_1u_{c1}$ ▷ 1M+1SR
7: $v_{c0} \leftarrow s_3 + s_1u_{c0}$, $\overline{E} \leftarrow [x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$ ▷ 1M
8: **return** $\overline{E}$ ▷ 4M+1S+2SR+1TR+1HT

3.4.4. Type III: $h(x) = 1$

In this section, the curve is of the form

$$C : y^2 + y = x^5 + f_3x^3 + f_1x + f_0 \tag{3.22}$$

with $f_0 \in \mathbb{F}_2$. According to the classification in Section 3.1, the 2-rank of Type-III curves is 0, i.e. the Picard group has odd order. In particular, this implies that every element in this group can be halved. Doing a case-by-case study of the doubling algorithm gives us the following theorem, where the halving formulas are obtained by inverting Cantor's algorithm (see Section 2.6.2).

Lemma 3.13. Let $\overline{D}_a = [u_a, v_a]$ be a divisor class in $\mathrm{Pic}^0(C)$ and let $\overline{D}_c = [u_c, v_c] = [\frac{1}{2}]\overline{D}_a$ be its preimage under the doubling.

(1) If $\deg(u_a) = 1$, then $\deg(u_c) = 2$ and we are in case HLV12.

(2) If $\deg(u_a) = 2$ and $u_{a1} \neq 0$, then $\deg(u_c) = 2$ and we are in case HLV22.

(3) If $\deg(u_a) = 2$ and $u_{a1} = 0$, then $u_c = x + \sqrt{u_{a0}}$, $v_c = v_{a0} + \sqrt{u_{a0}}v_{a1}$ and we are in case HLV21.

Proof. In (1), the degree of u_a is 1. According to Section 3.3.1, the only case with $\deg(u_a) = 1$ is HLV12.

In (2), the degree of u_a is 2 and $u_{a1} \neq 0$. Let us assume $\deg(u_c) = 1$, i.e. $u_c = x + u_{c0}$. Applying Cantor's algorithm to double $[u_c, v_c]$ gives $u_a = x^2 + u_{c0}^2$ with $u_{a1} = 0$, which is a contradiction. Hence $\deg(u_c) = 2$, and we are in the case HLV22.

In (3), we have $u_a = x^2 + u_{a0}$. Taking back Step 4 of Cantor's algorithm by extracting the square root of u_a, we get $u_c = x + \sqrt{u_{a0}}$, i.e. $\deg(u_c) = 1$, and we are in the case HLV21. Applying the combination step of Cantor's algorithm shows that $[2][u_c, v_c] = [x^2 + u_{a0}, v_a]$. We do not have to check if the preimage has odd order, because the Picard group of a Type-III curve has odd order. □

Algorithm 15 (HLV22, $h(x) = 1$, $f(x) = x^5 + f_3x^3 + f_1x + f_0$)

INPUT: The divisor class $\overline{D} = [x^2 + u_{a1}x + u_{a0}, v_{a1}x + v_{a0}]$
OUTPUT: The divisor class $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}] = [\frac{1}{2}]\overline{D}$

1: $s_0 \leftarrow \sqrt{u_{a1}}$, $s_1 \leftarrow s_0^{-1}$, $s_2 \leftarrow s_1 + f_3$, $u_{c1} \leftarrow \sqrt{s_2}$ ▷ 1I+2SR
2: $s_3 \leftarrow s_1\sqrt{u_{a0} + s_2}$, $v_{c1} \leftarrow \sqrt{s_3}$, $s_4 \leftarrow u_{a1}s_1$, $s_5 \leftarrow s_3 + s_4$ ▷ 2M+2SR
3: $s_6 \leftarrow u_{a0}s_5$, $s_7 \leftarrow 1 + v_{a0} + f_0 + s_6$, $v_{c0} \leftarrow \sqrt{s_7}$ ▷ 1M+1SR
4: $s_8 \leftarrow v_{a1} + f_1 + (u_{a0} + u_{a1})(s_1 + s_5) + s_4 + s_6$, $u_{c0} \leftarrow \sqrt{s_8}$ ▷ 1M+1SR
5: **return** $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$ ▷ 1I+4M+6SR

Algorithm 16 (HLV12, $h(x) = 1$, $f(x) = x^5 + f_3x^3 + f_1x + f_0$)

INPUT: The divisor class $\overline{D} = [x + u_{a0}, v_{a0}]$
OUTPUT: The divisor class $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}] = [\frac{1}{2}]\overline{D}$

1: $s_0 \leftarrow \sqrt{u_{a0}}$, $s_1 \leftarrow f_3$, $s_2 \leftarrow s_1u_{a0}$, $s_3 \leftarrow \sqrt{s_0 + s_2}$ ▷ 1M+2SR
2: $s_4 \leftarrow s_3 + f_1$, $s_5 \leftarrow v_{a0} + 1 + u_{a0}(s_3 + s_0u_{a0})$ ▷ 2M
3: $u_{c1} \leftarrow \sqrt{s_1}$, $u_{c0} \leftarrow \sqrt{s_4}$, $v_{c1} \leftarrow s_3 + \sqrt{s_2}$, $v_{c0} \leftarrow s_5 + s_0u_{c0}$ ▷ 1M+3SR
4: **return** $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$ ▷ 4M+5SR

Algorithm 17 (HLV21, $h(x) = 1$, $f(x) = x^5 + f_3x^3 + f_1x + f_0$)

INPUT: The divisor class $\overline{D}_a = [x^2 + u_{a0}, v_{a1}x + v_{a0}]$
OUTPUT: The divisor class $[x + u_{c0}, v_{c0}] = [\frac{1}{2}]\overline{D}_a$

1: $u_{c0} \leftarrow \sqrt{u_{a0}}$, $v_{c0} \leftarrow v_{a0} + u_{c0}v_{a1}$ ▷ 1M+SR
2: **return** $[x + u_{c0}, v_{c0}]$ ▷ 1M+1SR

3.4.5. Comparison

There are only a few published results on divisor class halving for hyperelliptic curves of genus 2 until now. The author in [Bir07] has considered curves of Type II and provided halving formulas that require 1I+8M+2S+5SR+1HT+1TR per halving. In this chapter, which is based on joint work with Nicolas Thériault [BT08], we improve the formulas in [Bir07] for Type II by 1SR. We do not explicitly list the operation count of the halving formulas in [Bir07] here.

In addition to [Bir07] and [BT08], there are halving formulas by Kitamura, Katagi and Takagi [KKT05], to which we compare the present work. In Table 3.1 we give the operation counts for divisor class halving for curves of Types Ia, Ic, II and III.

Type	$h(x)$	Kitamura et al. [KKT05]	This work (Section 3.4)
II	x	1I+15M+3S+3SR+2HT+2TR	1I+8M+2S+4SR+1HT+1TR
Ia	$x^2 + x + 1$	1I+14M+3S+3SR+2HT+2TR	1I+13M+2SR+2HT+2TR
Ic	x^2	1I+21M+2S+3SR+2HT+2TR	1I+12M+4SR+1HT+1TR
III	1	1I+21M+2S+3SR+2HT+2TR	1I+4M+6SR

Table 3.1.: Comparison of (worst case) operation counts for divisor class halving for different curves types

Note that Kitamura et al. in [KKT05] study some special cases, which cannot be obtained via generic isomorphic transformations.

3.5. Inversion-free arithmetic

The addition, doubling and halving formulas in the previous sections consist of a series of field multiplications, inversions, additions, squarings, square root extractions, trace and half-trace computations. Usually, the field inversion is the most expensive operation, followed by the field multiplication.

The term "I/M ratio" states how costly an inversion is compared to a multiplication. For instance, an I/M ratio of 20 means that 1 field inversion is as expensive as 20 field multiplications. The I/M ratio depends very much on how the field arithmetic is implemented. An important role plays the field library, the type of the CPU and if one is doing a software or hardware implementation.

Since the formulas in this chapter are designed for fields of characteristic 2, they are especially useful for hardware implementations. Usually, for those applications inversions should be completely avoided as the I/M ratio is usually large. This is due to higher production costs of hardware inverters compared to multiplication or addition units. To resolve this problem, we present addition and doubling formulas without any field inversion at the cost of slightly higher number of field multiplications.

The main contributions of this section are the doubling formulas in recent coordinates in Section 3.5.2 which improve the fastest ones, published in [ACD+05], Algorithm 14.51.

3.5.1. New coordinates

In this section, we give inversion-free addition and inversion-free doubling formulas in new coordinates for hyperelliptic curves of Type II, i.e. the curve equation is of the form

$$C : y^2 + xy = x^5 + f_3x^3 + f_2x^2 + f_0, \tag{3.23}$$

where $f_2 \in \mathbb{F}_2$.

To remove all field inversions in the formulas, Lange (see Section 6 in [Lan05] and Section 14.5.4.a in [ACD+05]) suggests to let $[U_1, U_0, V_1, V_0, Z_1, Z_2]$ correspond to the affine divisor class

$$[x^2 + U_1/Z_1^2x + U_0/Z_1^2, V_1/(Z_1^3Z_2)x + V_0/(Z_1^3Z_2)]. \tag{3.24}$$

This representation is called *new coordinates*.

The following doubling formulas are taken from Appendix 2 in [Lan05]. Note that in addition to the 6 coordinates, we use the four cached values $z_1 = Z_1^2$, $z_2 = Z_2^2$, $z_3 = Z_1Z_2$, $z_4 = Z_1^3Z_2$ for the input and output. If $h_1 = 1$, then the operation count decreases from 35M+6M to 30M+6S.

Algorithm 18 (DBL22, $h(x) = x$, $f(x) = x^5 + f_3x^3 + f_2x^2 + f_0$, $f_2 \in \mathbb{F}_2$)

INPUT: The divisor class $\overline{D} = [U_1, U_0, V_1, V_0, Z_1, Z_2, z_1, z_2, z_3, z_4]$

OUTPUT: The divisor class $[U_1', U_0', V_1', V_0', Z_1', Z_2', z_1', z_2', z_3', z_4'] = [2]\overline{D}$

1: $w_1 \leftarrow h_1U_1 + h_0z_1$, $r \leftarrow h_0w_1 + h_1^2U_0$, $\tilde{Z}_2 \leftarrow rz_4$, $Z_2' \leftarrow \tilde{Z}_2z_4$ ▷ 6M

2: $\text{inv}_1 \leftarrow h_1$, $\text{inv}_0 \leftarrow w_1$, $w_0 \leftarrow V_1^2$, $w_1 \leftarrow U_1^2$ ▷ 2S

3: $k_1 \leftarrow z_2(f_3 z_1^2 + w_1)$, $k_0 \leftarrow U_1 k_1 + w_0 + z_4(f_2 z_4 + V_1 h_1)$ ▷ 5M+1S
4: $w_0 \leftarrow k_0 \mathrm{inv}_0$, $w_1 \leftarrow k_1 \mathrm{inv}_1$ ▷ 2M
5: $s_1 \leftarrow (\mathrm{inv}_0 + \mathrm{inv}_1)(k_0 + k_1) + w_0 + w_1(1 + U_1)$ ▷ 2M
6: $s_0 \leftarrow w_0 + U_0 w_1 z_1$, $Z_1' \leftarrow s_1 z_1$, $S_0 \leftarrow s_0^2$, $S \leftarrow s_0 Z_1'$ ▷ 4M+1S
7: $R \leftarrow \tilde{Z}_2 Z_1'$, $z_1' \leftarrow Z_1'^2$, $z_2' \leftarrow {Z_2'}^2$, $z_3' \leftarrow Z_1' Z_2'$, $z_4' \leftarrow z_1' z_3'$ ▷ 3M+2S
8: $s_0 \leftarrow s_0 s_1$, $s_1 \leftarrow Z_1' s_1$, $\tilde{h}_1 \leftarrow h_1 z_3'$ ▷ 3M
9: $l_2 \leftarrow s_1 U_1$, $l_0 \leftarrow s_0 U_0$, $l_1 \leftarrow (s_1 + s_0)(U_1 + U_0) + l_0 + l_2$ ▷ 3M
10: $l_2 \leftarrow l_2 + S$, $U_0' \leftarrow S_0 + \tilde{h}_1$, $U_1' \leftarrow z_2'$
11: $l_2 \leftarrow l_2 + U_1'$, $w_0 \leftarrow l_2 U_0'$, $w_1 \leftarrow l_2 U_1'$ ▷ 2M
12: $V_1' \leftarrow w_1 + z_1'(l_1 + R V_1 + U_0' + \tilde{h}_1)$ ▷ 2M
13: $V_0' \leftarrow w_0 + z_1'(l_0 + R V_0 + z_3' h_0)$ ▷ 3M
14: **return** $[U_1', U_0', V_1', V_0', Z_1', Z_2', z_1', z_2', z_3', z_4']$
▷ Worst case: 35M+6S (30M+6 if $h_1 = 1$)

The following addition formulas are taken from Appendix 2 in [Lan05]. As explained in the same appendix, to save space we can discard the coordinates Z_1 and Z_2. The values z_1, z_2, z_3, z_4 are again cached values.

Algorithm 19 (ADD, $h(x) = x$, $f(x) = x^5 + f_3 x^3 + f_2 x^2 + f_0$, $f_2 \in \mathbb{F}_2$)

INPUT: Two divisor classes $\overline{D}_1 = [U_{11}, U_{10}, V_{11}, V_{10}, z_{11}, z_{12}, z_{13}, z_{14}]$ and $\overline{D}_2 = [U_{21}, U_{20}, V_{21}, V_{20}, z_{21}, z_{22}, z_{23}, z_{24}]$
OUTPUT: The divisor class $[U_1', U_0', V_1', V_0', z_1', z_2', z_3', z_4'] = \overline{D}_1 \oplus \overline{D}_2$

1: $\tilde{U}_{21} \leftarrow U_{21} z_{11}$, $\tilde{U}_{20} \leftarrow U_{20} z_{11}$, $\tilde{V}_{21} \leftarrow V_{21} z_{14}$, $\tilde{V}_{20} \leftarrow V_{20} z_{14}$ ▷ 4M
2: $Z_1 \leftarrow z_{11} z_{21}$, $Z_3 \leftarrow z_{13} z_{23}$, $y_1 \leftarrow U_{11} z_{21} + \tilde{U}_{21}$ ▷ 3M
3: $y_2 \leftarrow U_{10} z_{21} + \tilde{U}_{20}$, $y_3 \leftarrow U_{11} y_1 + y_2 z_{11}$, $r \leftarrow y_2 y_3 + y_1^2 U_{10}$ ▷ 5M+1S
4: $\tilde{Z}_2 \leftarrow r Z_3$, $Z_2' \leftarrow \tilde{Z}_2 Z_1$, $\tilde{Z}_2 \leftarrow \tilde{Z}_2^2$, $\tilde{Z}_2 \leftarrow \tilde{Z}_2 Z_1$ ▷ 3M+1S
5: $\mathrm{inv}_1 \leftarrow y_1$, $\mathrm{inv}_0 \leftarrow y_3$, $w_0 \leftarrow V_{10} z_{24} + \tilde{V}_{20}$ ▷ 1M
6: $w_1 \leftarrow V_{11} z_{24} + \tilde{V}_{21}$, $w_2 \leftarrow \mathrm{inv}_0 w_0$, $w_3 \leftarrow \mathrm{inv}_1 w_1$ ▷ 3M
7: $s_1 \leftarrow (\mathrm{inv}_0 + \mathrm{inv}_1 z_{11})(w_0 + w_1) + w_2 + w_3(z_{11} + U_{11})$ ▷ 3M
8: $s_0 \leftarrow w_2 + w_3 U_{10}$ ▷ 1M
9: $S_1 \leftarrow s_1^2$, $Z_1' \leftarrow s_1 Z_1$, $R \leftarrow r Z_1'$, $S_0 \leftarrow s_0 Z_1$ ▷ 3M+1S
10: $S \leftarrow S_0 Z_1'$, $S_0 \leftarrow S_0^2$, $z_1' \leftarrow Z_1'^2$, $z_2' \leftarrow Z_2'^2$, $z_3' \leftarrow Z_1' Z_2'$ ▷ 2M+3S
11: $z_4' \leftarrow z_1' z_3'$, $\tilde{h}_1 \leftarrow h_1 z_3'$, $s_1 \leftarrow s_1 Z_1'$, $s_0 \leftarrow s_0 Z_1'$ ▷ 4M
12: $l_2 \leftarrow s_1 \tilde{U}_{21}$, $l_0 \leftarrow s_0 \tilde{U}_{20}$, $l_1 \leftarrow (s_0 + s_1)(\tilde{U}_{20} + \tilde{U}_{21}) + l_2 + l_0$ ▷ 3M

13: $l_2 \leftarrow l_2 + S$, $U_0' \leftarrow S_0 + y_1(S_1(y_1 + \tilde{U}_{21}) + \tilde{Z}_2) + y_2 s_1 + \tilde{h}_1$ ▷ 3M
14: $U_1' \leftarrow y_1 s_1 + z_2'$, $l_2 \leftarrow l_2 + U_1'$, $w_0 \leftarrow l_2 U_0'$, $w_1 \leftarrow l_2 U_1'$ ▷ 3M
15: $V_1' \leftarrow w_1 + z_1'(l_1 + R\tilde{V}_{21} + U_0' + \tilde{h}_1)$ ▷ 2M
16: $V_0' \leftarrow w_0 + z_1'(l_0 + R\tilde{V}_{20}) + z_4' h_0$ ▷ 3M
17: **return** $[U_1', U_0', V_1', V_0', Z', z']$ ▷ Total: 46M+6S
▷ Mixed addition (one input is affine): 38M+6S

3.5.2. Recent coordinates

In this section, we work with hyperelliptic curves of Type II over $\mathbb{F}_{2^d}$ for an odd integer d, i.e. the curve equation is

$$C : y^2 + h_1 xy = x^5 + f_3 x^3 + f_2 x^2 + f_0, \tag{3.25}$$

where $f_2 \in \mathbb{F}_2$ and $h_1, f_3, f_0 \in \mathbb{F}_{2^d}$. Type II normally requires $h_1 = 1$, but the addition and doubling formulas in this section are slightly more general and allow arbitrary values for h_1. We give separate operation counts for $h_1 = 1$.

In 2005, Lange (cf. Section 14.5.5.b in [ACD+05]) suggests to let $[U_1, U_0, V_1, V_0, Z]$ correspond to the affine divisor class

$$[x^2 + U_1/Z x + U_0/Z, V_1/Z^2 x + V_0/Z^2]. \tag{3.26}$$

This representation is called *recent coordinates*. In the following, we propose new doubling formulas in recent coordinates for curves of Type II.

Algorithm 20 (DBL22, $h(x) = h_1 x$, $f(x) = x^5 + f_3 x^3 + f_2 x^2 + f_0$)

INPUT: A divisor class $\overline{D} = [U_1, U_0, V_1, V_0, Z]$ in recent coordinates and the precomputed values h_1^2, h_1^{-1}
OUTPUT: The doubled divisor class $[U_1', U_0', V_1', V_0', Z'] = [2]\overline{D}$

1: $Z_2 \leftarrow Z^2$, $Z_4 \leftarrow {Z_2}^2$, $t_1 \leftarrow f_0 Z_4 + V_0^2$, $t_2 \leftarrow U_1^2 + f_3 Z_2$ ▷ 2M+4S
2: $a_1 \leftarrow U_0^2$, $a_2 \leftarrow a_1 Z$, $a_3 \leftarrow h_1^2 a_1 Z_4$, $q_1 \leftarrow (t_2 a_2 + U_1 t_1)^2 + t_1 a_3$ ▷ 6M+2S
3: $q_2 \leftarrow t_1^2$, $q_3 \leftarrow q_2^2$, $q_4 \leftarrow a_1 Z_2$, $q_5 \leftarrow t_1 t_2$, $q_6 \leftarrow (a_3 + q_5) t_1$ ▷ 3M+2S
4: $U_0' \leftarrow q_1$, $U_1' \leftarrow a_3 q_4$, $V_0' \leftarrow h_1^{-1}(q_6 q_1 + q_3 q_4)$ ▷ 4M
5: $V_1' \leftarrow h_1^{-1}\Big(q_4(q_5 q_6 + a_3^2 t_1) + q_3(f_2 Z_4 + V_1^2)\Big)$, $Z' \leftarrow q_2 Z_2$ ▷ 7M+2S
6: **return** $[U_1', U_0', V_1', V_0', Z']$ ▷ 22M+10S (19M+10S if $h_1 = 1$)

The following addition formulas in recent coordinates for curves of Type II are taken from Algorithm 14.50 in [ACD+05]. The input can be either two divisor

classes in recent coordinates, or one divisor class in recent coordinates and the other one in affine coordinates. In the latter case, either Z_1 or Z_2 is equal to 1. An addition in this way is called *mixed addition* and the operation count decreases from 50M+8S to 43M+7S if $Z_1 = z_1 = 1$.

Algorithm 21 (ADD, $h(x) = h_1 x$, $f(x) = x^5 + f_3 x^3 + f_2 x^2 + f_0$)

INPUT: Two divisor classes $\overline{D}_1 = [U_{11}, U_{10}, V_{11}, V_{10}, Z_1, z_1]$ and $\overline{D}_2 = [U_{21}, U_{20}, V_{21}, V_{20}, Z_2, z_2]$

OUTPUT: The divisor class $[U_1', U_0', V_1', V_0', Z', z'] = \overline{D}_1 \oplus \overline{D}_2$

1: $Z \leftarrow Z_1 Z_2$, $z \leftarrow Z^2$, $\tilde{U}_{21} \leftarrow U_{21} Z_1$, $\tilde{U}_{20} \leftarrow U_{20} Z_1$ ▷ 3M+1S
2: $\tilde{V}_{21} \leftarrow V_{21} z_1$, $\tilde{V}_{20} \leftarrow V_{20} z_1$ ▷ 2M
3: $y_1 \leftarrow U_{11} Z_2 + \tilde{U}_{21}$, $y_2 \leftarrow U_{10} Z_2 + \tilde{U}_{20}$ ▷ 2M
4: $y_3 \leftarrow U_{11} y_1 + y_2 Z_1$, $r \leftarrow y_2 y_3 + y_1^2 U_{10}$ ▷ 4M+1S
5: $w_0 \leftarrow V_{10} z_2 + \tilde{V}_{20}$, $w_1 \leftarrow V_{11} z_2 + \tilde{V}_{21}$ ▷ 2M
6: $w_2 \leftarrow y_3 w_0$, $w_3 \leftarrow y_1 w_1$ ▷ 2M
7: $s_1 \leftarrow (y_3 + y_1 Z_1)(w_0 + w_1) + w_2 + w_3 (Z_1 + U_{11})$ ▷ 3M
8: $s_0 \leftarrow w_2 + U_{10} w_3$ ▷ 1M
9: $\overline{Z} \leftarrow s_1 r$, $w_4 \leftarrow r Z$, $w_5 \leftarrow w_4^2$, $S \leftarrow s_0 Z$, $Z' \leftarrow Z \overline{Z}$ ▷ 4M+1S
10: $\tilde{s}_0 \leftarrow s_0 Z'$, $\overline{s}_1 \leftarrow s_1 \overline{Z}$, $\tilde{s}_1 \leftarrow \overline{s}_1 Z$ ▷ 3M
11: $L_2 \leftarrow \overline{s}_1 \tilde{U}_{21}$, $\ell_2 \leftarrow L_2 Z$, $\ell_0 \leftarrow \tilde{s}_0 \tilde{U}_{20}$ ▷ 3M
12: $\ell_1 \leftarrow (\tilde{U}_{21} + \tilde{U}_{20})(\tilde{s}_0 + \tilde{s}_1) + \ell_2 + \ell_0$ ▷ 1M
13: $\ell_2 \leftarrow L_2 + \tilde{s}_0$, $\tilde{h}_1 \leftarrow h_1 z$ ▷ 1M
14: $U_0' \leftarrow r(S^2 + y_1(s_1^2(y_1 + \tilde{U}_{21}) + Z w_5) + \tilde{h}_1 Z') + y_2 \tilde{s}_1$ ▷ 6M+2S
15: $U_1' \leftarrow y_1 \overline{s}_1 + w_4 w_5$ ▷ 2M
16: $w_1 \leftarrow \ell_2 + U_1'$, $U_1' \leftarrow U_1' w_4$, $\overline{Z} \leftarrow Z' \overline{Z}$, $\ell_0 \leftarrow \ell_0 \overline{Z}$ ▷ 3M
17: $w_2 \leftarrow U_1' w_1 + (U_0' + \ell_1)\overline{Z}$, $\overline{Z} \leftarrow \overline{Z}^2$ ▷ 2M+1S
18: $V_1' \leftarrow w_2 s_1 + (\tilde{V}_{21} + \tilde{h}_1)\overline{Z}$, $U_0' \leftarrow U_0' r$, $w_2 \leftarrow U_0' w_1 + \ell_0$ ▷ 4M
19: $V_0' \leftarrow w_2 s_1 + \tilde{V}_{20} \overline{Z}$, $Z' \leftarrow Z'^2$, $z' \leftarrow Z'^2$ ▷ 2M+2S
20: **return** $[U_1', U_0', V_1', V_0', Z', z']$ ▷ Worst case: 50M+8S
▷ Mixed addition ($Z_1 = z_1 = 1$): 43M+7S

If $h_1 = 1$, then one more multiplication can be saved in Step 13.

Note that the inputs and the output of the addition formulas are given in recent coordinates, but they include a sixth value z_i, which is not an additional coordinate, but the cached value $z_i = Z_i^2$. This saves 1 field squaring per addition.

If it is important to use as little memory as possible, then those values can be omitted at the cost of 1 additional field squaring per addition.

3.6. Choice of secure curves

In the previous sections we have studied hyperelliptic curves of genus 2 over binary fields, and provided efficient arithmetic on these curves. Now we will discuss the security of these curves and suggest curve parameters and field sizes to make the DLP on these curves intractable against currently known attacks.

For genus 2 index calculus attacks (see Appendix A.4) are not applicable. To avoid the other types of attacks we restrict the base field to $\mathbb{F}_{2^d}$, where d is prime (this is to avoid Weil descent attacks, see Section A.6 for details), request that the embedding degree k is at least 2000 (to avoid Frey-Rück attacks, see Section A.5) and that the order of $\mathrm{Pic}^0_{\mathbb{F}_{2^d}}(C)$ is almost prime (i.e. it is of the form cp, where p is a prime number and c a small integer that is called *cofactor*) and sufficiently large. Due to the Hasse-Weil bound (Theorem 2.15) the size of the Picard group of a hyperelliptic curve of genus 2 over $\mathbb{F}_{2^d}$ is at most $(\sqrt{2^d}+1)^4$ which is approximately 2^{2d}. Since the best known attacks for genus 2 are square-root attacks, e.g. baby-step giant-step (Section A.2) and Pollard rho (Section A.3), the security level of these curves is d bit.

According to the latest recommendations 80-bit security (i.e. $d \approx 80$) is borderline and 112 bit offer medium-term protection whereas 128 bit are good for long-term secrecy (see [Gir08] and the ECRYPT recommendations [Nae08] for more details). Suitable prime numbers in these ranges are $d = 83$, $d = 113$ and $d = 131$.

3.6.1. HECTOR

To illustrate these choices we provide a secure curve which we also used for an implementation. HECTOR (Hyperelliptic Curve with Two-Rank One) uses a hyperelliptic curve of genus 2 with security level 113 bit and satisfying the requirements of the previous section. This implementation is joint work with Peter Schwabe. We implemented the Diffie-Hellman key exchange protocol (see Section 1.6.1 in [ACD+05]) and an ElGamal signature scheme using that curve. The equation of the HECTOR curve is

$$C : y^2 + xy = x^5 + t^{55}x^3 + x^2 + t^{53}, \tag{3.27}$$

and the curve is defined over the field $\mathbb{F}_2[t]/(t^{113} + t^9 + 1)$. (We would like to thank Wouter Castryck, Katholieke Universiteit Leuven, Belgium for providing this curve.)

The finite field implementation

The implementation of the finite field is based on the $\texttt{mp}\mathbb{F}_q$ library [GT07]. More than just a library, $\texttt{mp}\mathbb{F}_q$ is a code generator for finite field arithmetic. Algorithms for reduction can hence be optimised for just one special finite field. The library makes extensive use of the SSE2 processor extensions, so HECTOR will only run on machines, where this extension is available.

Scalar multiplication

We used recent coordinates (see Section 3.5.2) for the group operation. For the scalar multiplication we use two different algorithms: To compute multiples of the generator we use the 2-table comb method with 512 precomputed multiples in total. For the computation of multiples of points other than the generator we use the windowed NAF method with a window size of 5.

Performance

The HECTOR implementation was submitted to the ECRYPT benchmarking project eBACS (ECRYPT Benchmarking of Cryptographic Systems, see [Be]). The key exchange and signature implementation were benchmarked for time and key size on a variety of machines (e.g. Intel Core 2 Duo, AMD Athlon 64 X2, Intel Pentium D, Intel Core 2 Quad, Intel Pentium M).

For example, performing the Diffie-Hellman key-exchange protocol on an Intel Core 2 Quad (6f8, "nmiv004") we get the following figures for HECTOR and surf2113. Both systems are based on a genus-2 hyperelliptic curve over a binary field with 113 bit. The lengths of the secret key, public key and of the shared secret are 29 byte, 60 byte and 60 byte for HECTOR, and 28 byte, 48 byte and 48 byte byte for surf2113.

	Key pair generation	Shared secret computation
HECTOR	523647	1729962
surf2113	1184571	1163061
Percentage	44.21%	148.74%

Table 3.2.: Diffie-Hellman benchmarks of HECTOR and surf2113 (in CPU cycles and percent)

The surf2113 implementation uses a representation that does not allow general additions. Therefore, no corresponding signature scheme exists. Instead, we compare HECTOR to 1024-bit DSA using OpenSSL on an Intel Pentium 4 (f41, "svlin002") as reported by the same benchmarking project.

	Key pair generation	Signing a short message	Verification
HECTOR	1031805	1122953	4644240
1024-bit DSA	12486750	11816250	14106720
Percentage	8.26%	9.50%	32.92%

Table 3.3.: Signature scheme benchmarks of HECTOR and 1024-bit DSA (in CPU cycles and percent)

Observe that the security level of 1024-bit DSA is at most 80 bit, while HECTOR was designed to have 113 bit security.

3.6.2. More secure curves

In this section we give a list of hyperelliptic curves of genus 2 over $\mathbb{F}_{2^d}$, where d is 83, 113, 149 and 163 bit. The security level (see Section 3.6) of these curves is also 83, 113, 149 and 163 bit. All curves are of Type II, i.e. the curve equation is

$$C : y^2 + xy = x^5 + f_3x^3 + f_2x^2 + f_0, \tag{3.28}$$

where $f_2 \in \mathbb{F}_2$. In the following four lists, the polynomial f is the right-hand side of (3.28) and N the order of the Picard group of $C/\mathbb{F}_{2^d}$, which is always of the form $N = 2p$ for a prime number p.

For each extension field we state the polynomial $w \in \mathbb{F}_2[t]$, which was used to construct $\mathbb{F}_{2^d}$ as $\mathbb{F}_2[t]/(w)$. This polynomial is primitive over $\mathbb{F}_2$ in all four cases, and since $w(t) = 0$ in $\mathbb{F}_{2^d}$, the value t is a primitive element of $\mathbb{F}_{2^d}$. Hence we can give the coefficients of the polynomial f as powers of t.

All curves were found using a Magma ([BCP97]) script, which chooses random values in $\mathbb{F}_{2^d}$ for f_3 and f_0 and calculates the order of the Picard group of $C/\mathbb{F}_{2^d}$. We always use $f_2 = 1$ to force the Picard group to have order $2r$, where r is odd (see Corollary 3.5).

Type-II curves over $\mathbb{F}_{2^{83}}$

We use the defining polynomial $w = z^{83} + z^7 + z^4 + z^2 + 1$ for all 83-bit curves.

(1) $f = x^5 + t^{54}x^3 + x^2 + t^{29}$
$N = 2 \cdot 4676805239456175154235478466098724977974536298817 7$

(2) $f = x^5 + t^{69}x^3 + x^2 + t^{33}$
$N = 2 \cdot 4676805239457730824694926696792724074033524866655 3$

(3) $f = x^5 + t^{23}x^3 + x^2 + t^{31}$
$N = 2 \cdot 4676805239460740676594958167669647366402173100524 9$

(4) $f = x^5 + tx^3 + x^2 + t^{10}$
$N = 2 \cdot 467680523945793371941838283928245942594969912764 41$

(5) $f = x^5 + t^4x^3 + x^2 + t^{77}$
$N = 2 \cdot 467680523945795434677985441906380086696425199777 53$

(6) $f = x^5 + t^7x^3 + x^2 + t^{27}$
$N = 2 \cdot 467680523945817834494082573236151003081244771262 21$

(7) $f = x^5 + t^{47}x^3 + x^2 + t^{54}$
$N = 2 \cdot 467680523945754984072172196302000036632624359878 01$

(8) $f = x^5 + t^4x^3 + x^2 + t^{16}$
$N = 2 \cdot 467680523945685649843639290977912435169612311338 09$

(9) $f = x^5 + t^{78}x^3 + x^2 + t^4$
$N = 2 \cdot 467680523946299159161642285622882649212515675048 09$

(10) $f = x^5 + t^{26}x^3 + x^2 + t^{46}$
$N = 2 \cdot 467680523945932678157201690057036511954155931350 81$

Type-II curves over $\mathbb{F}_{2^{113}}$

We use the defining polynomial $w = z^{113} + z^5 + z^3 + z^2 + 1$ for all 133-bit curves.

(1) $f = x^5 + t^{40}x^3 + x^2 + t^{31}$
$N = 2 \cdot 5391989333430128045543337571092281087923275559872880780160246375112 7$

(2) $f = x^5 + t^{95}x^3 + x^2 + t^{58}$
$N = 2 \cdot 5391989333430127974088330373466052812040822168550183060618727529936 9$

(3) $f = x^5 + t^{22}x^3 + x^2 + t^{90}$
$N = 2 \cdot 5391989333430127927457290854218264713174056243832340495056913793561 7$

Type-II curves over $\mathbb{F}_{2^{149}}$

We use the defining polynomial $w = z^{149} + z^9 + z^7 + z^6 + z^5 + z^4 + z^3 + z + 1$ for all 149-bit curves.

(1) $f = x^5 + t^{34}x^3 + x^2 + t^{132}$
$N = 2 \cdot 2546294970418107607835438378417588635273942855856874157446929314241470164749426552259893 17$

(2) $f = x^5 + t^6x^3 + x^2 + t^{22}$
$N = 2 \cdot 254629497041810760783571353585685813581160610958295629$
$251155448003340361494968858668726433$

(3) $f = x^5 + t^{20}x^3 + x^2 + t^{37}$
$N = 2 \cdot 254629497041810760783557207378766737922615296764621184$
$516547749621154313446393714379530769$

(4) $f = x^5 + tx^3 + x^2 + t^{96}$
$N = 2 \cdot 254629497041810760783551696533799163638182445019953900$
$369519745891403966840037729120940689$

Type-II curves over $\mathbb{F}_{2^{163}}$

We use the defining polynomial $w = z^{163} + z^7 + z^6 + z^3 + 1$ for all 163-bit curves.

(1) $f = x^5 + t^{118}x^3 + x^2 + t^{126}$
$N = 2 \cdot 68351585149469122636640660158735336964615572224543393 1$
$306954742495981607159260498591632638598366 73$

(2) $f = x^5 + t^{31}x^3 + x^2 + t^{83}$
$N = 2 \cdot 68351585149469122636640728566987817255386373954650362 5$
$263855097230612528961486106870141738868244 49$

(3) $f = x^5 + t^{132}x^3 + x^2 + t^{98}$
$N = 2 \cdot 68351585149469122636640719786728639696531352877772404 3$
$708429224184623359085238973683354757249059 53$

4. Arithmetic on Genus-3 Curves over Binary Fields

In this chapter, we investigate halving and doubling of divisor classes of hyperelliptic curves of genus 3 over finite fields of characteristic 2. With this, we extend the results on genus-2 halving in Chapter 3. The work in this chapter is joint work with Nicolas Thériault.

In the following sections, we present efficient halving and doubling formulas for many interesting cases (i.e. different types of curves). In Section 4.1, we explain why it is important to use special types of binary fields (e.g. fields with prime extension degree to avoid Weil descent attacks A.6). In Section 4.2 we discuss the order of the Picard group of the curves, in particular if it is odd or 2 times an odd number. In the next section we sort hyperelliptic curves of genus 3 into different categories, depending on the degree and factorisation of the polynomial h in the curve equation. In the following section we list the different types of curves and give their main properties. In Section 4.4, we provide isomorphic transformations to each type of curve and achieve a more simple form for each case. This is mainly for performance reasons. In Section 4.5, we investigate the optimal-performance case, i.e. we obtain the best operation counts for the explicit doubling and halving formulas. In the same section, we also give a complete case study for the most frequent case and for all special cases which can occur when doubling or halving a divisor class. This provides a programmer with everything for a complete implementation of high-speed scalar multiplication. In Section 4.6, we treat other interesting cases. In particular, we look at curves the equation of which has a polynomial h that is different from 1. Those cases are especially interesting since we gain comparable and sometimes even noticeable better performance for the halving compared to the appropriate doubling formulas. For these cases, we give explicit halving formulas for the most common case.

In a normal double-and-add scalar multiplication, all but an almost insignificant proportion of the additions and doublings should fall in the most common cases. One can then implement explicit formulas only for the most common cases, and use Cantor's algorithm (see Section 2.6.2) when a special case occurs. In practise, this approach does not create any measurable loss in the average performance compared to an implementation which has explicit formulas for all possible cases. The same is not so clearly true for the halve-and-add algorithm. However, since the inverse operation of Cantor's doubling algorithm cannot easily be written in terms of polynomials, a halve-and-add algorithm must therefore contain explicit

formulas for all possible cases of the halving operation.

Like in Chapter 3, we always work with the Mumford representation (see Theorem 2.16) of a divisor class and obtain the different cases depending on the degree of the first polynomial of the Mumford representation of the inputs and outputs.

Main results

The main results of this chapter are as follows:

(1) For genus-3 curves with $h(x) = 1$ (this is the optimal-performance case), we provide explicit doubling formulas for all special cases, and we thereby extend the formulas which are already published for the most common case only [ATW08, FWW06, GKP04].

(2) In the optimal-performance case, we also provide explicit halving formulas for all possible cases and therefore allow a complete implementation of a DLP-based cryptosystem on a genus-3 curve using halving (and doubling) of divisor classes.

(3) We look at three more general types of genus-3 hyperelliptic curves and provide halving formulas which compare extremely well to the best previously known doubling formulas. It turns out that in those cases halving is always faster. In some situations halving is almost twice as fast as the corresponding doubling operation.

4.1. Choice of the field and divisor class halving

Throughout this chapter, we will assume that the field is $\mathbb{F}_{2^d}$, where d is odd and not divisible by 3. This is mainly due to security concerns, since various versions of the Weil descent attack (see Appendix A.6) could be applied when d admits a factor of 2 or 3 (for example, see [Gau04, GHS02, Thé03b]). In fact, for cryptographic applications it is often assumed that d is a prime number. As an added bonus, having d coprime to 6 means that we can take third, fifth and seventh roots in the field (since the map $\alpha \mapsto \alpha^i$ for $i = 3, 5, 7$ is an isomorphism as 3, 5 and 7 are coprime to $2^d - 1$), which allows us to simplify the curve equations a little more.

In finite fields of characteristic 2, some operations which are computationally hard in fields of odd characteristic become efficient, in particular the computation of the square root of a field element. This observation led to the development of halve-and-add algorithms, a variation of the double-and-add scalar multiplication where the doubling operation is replaced with a halving (the representation of the scalar is adjusted accordingly). Such an approach was first used for elliptic curves [Knu99, Sch00b], and was recently extended to hyperelliptic curves of

genus 2 (see Chapter 3 and [KKT05, Bir07, BT08]). In fact, some fields have the property that the computation of square roots can be faster than the computation of squares [AT07, FHLM04]. It can therefore become a good strategy to "replace" squares with square roots for curve arithmetic in these fields, and this is exactly what our halving formulas do. Furthermore, since d will be odd, we will have $\mathrm{TR}(1) = 1$. In various places, we implicitly take advantage of the identity $\mathrm{TR}(\alpha) = \mathrm{TR}(\alpha^2)$ (see (3) in Lemma 3.3) to simplify some trace computations.

To count the number of operations, we denote inversions by I, multiplications by M, squares by S, square roots by SR, traces by TR and half-traces by HT.

4.2. Conditions on the order of the Picard group

We limit ourselves to curves for which the order of the Picard group is either odd ($h(x)$ constant) or 2 times an odd number. This restriction is needed to get a better performance out of the halving. Given any hyperelliptic curve, the halve-and-add algorithm allows us to compute the scalar multiple of a divisor class, given that it is in a (sub)group of odd order. In this way, the preimage of the doubling can always be computed and "becomes" unique (all other preimages of the doubling have even order). The group order conditions are due to the following reasons:

(1) To verify that the preimage is in the subgroup of odd order, we make sure that it can be halved again as many times as we want. If the group contains divisor classes of order 2^r, we use as a test criterion that we can halve the preimage (at least) r times, which obviously affects the cost of our halving formulas. When $r \geq 2$ (e.g. when there is a divisor class of order 4), the increased work required for this check becomes too expensive for the halving to be interesting.

(2) The number of preimages of the halving depends directly on the number of divisor classes of order 2 in the group, which in turn depends on the factorisation of $h(x)$. If $h(x)$ has r distinct irreducible factors (multiplicities do not have an impact here), then we have 2^r distinct preimages of the doubling. Since we must identify the unique preimage of odd order, having $r > 1$ would force us to choose between four or more divisor classes, which increases the algorithmic cost of halving significantly. We will therefore require r to be at most 1.

Note that if $h(x)$ has r distinct irreducible factors, then the group order is divisible by (at least) 2^r, so asking the group order to be either odd or 2 times an odd number removes all curves for which $h(x)$ has 2 or 3 distinct irreducible factors.

4.3. Types of curves

We can distinguish the genus-3 hyperelliptic curves in characteristic 2 according to the degree of $h(x)$ and the form of its factorisation over $\mathbb{F}_{2^d}$. We find the following types:

- Type Ia: $h(x)$ is irreducible of degree 3.
- Type Ib: $h(x)$ has degree 3 and is the product of an irreducible polynomial of degree 2 and a linear factor ($r = 2$).
- Type Ic: $h(x)$ has degree 3 and is the product of 3 distinct linear factors ($r = 3$).
- Type Id: $h(x)$ has degree 3 and is the product of 2 distinct linear factors, one of which is repeated twice ($r = 2$).
- Type Ie: $h(x)$ is the cube of a linear factor (degree 3, $r = 1$).
- Type IIa: $h(x)$ is irreducible of degree 2.
- Type IIb: $h(x)$ has degree 2 and is the product of 2 distinct linear factors ($r = 2$).
- Type IIc: $h(x)$ is the square of a linear factor (degree 2, $r = 1$).
- Type III: $h(x)$ is linear (degree 1).
- Type IV: $h(x)$ is constant (degree 0).

For each type of curve, we can use curve isomorphisms to "simplify" the equation of the curve. This will be handled in the next subsection.

Due to our condition on the group order, we will limit ourselves to curves of Types Ia, IIa, III and IV. Because of the structure of their 2-torsion group, curves of Types Ie and IIc have very similar properties (and essentially the same number of isomorphism classes) as curves of Type III. On the other hand, the higher degree of $h(x)$ in Type Ie and IIc makes them less efficient than curves of Type III.

4.4. Forms of the curve equations

An imaginary hyperelliptic curve of genus 3 over $\mathbb{F}_{2^d}$ is of the form

$$C : y^2 + h(x)y = f(x), \tag{4.1}$$

where $h(x) = h_3x^3 + h_2x^2 + h_1x + h_0 \neq 0$ and $f(x) = f_7x^7 + f_6x^6 + f_5x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$. It is customary to use isomorphisms to impose that $f(x)$

is monic, i.e. that $f_7 = 1$, but we will relax this condition for some curves as the halving formulas are more efficient if we use the isomorphisms to force a specific coefficient of $h(x)$ to be 1 (which coefficient depends on the curve type).

For a curve given by (4.1), the possible isomorphisms are

$$x \mapsto \alpha x + \beta \quad \text{and} \quad y \mapsto \gamma y + \delta x^3 + \varepsilon x^2 + \varrho x + \zeta, \tag{4.2}$$

where both α and γ are non-zero. After applying the isomorphisms, the equation is divided by γ^2 to make it monic.

Proposition 4.1. Given an isomorphism that replaces f_{2i} by $f_{2i} + \omega^2 + \omega$, then we can restrict f_{2i} to $\mathrm{TR}(f_{2i}) \in \mathbb{F}_2$.

Proof. Since $f_{2i} + \mathrm{TR}(f_{2i})$ has trace 0, we can choose ω such that $\omega^2 + \omega = f_{2i} + \mathrm{TR}(f_{2i})$. This choice of ω replaces f_{2i} with $\mathrm{TR}(f_{2i})$. Note that the isomorphism does not permit us to change the trace of f_{2i}. □

For the four types of curves where halving is interesting, we have:

(Ia) $h_3 \neq 0$ and $h(x)$ irreducible:

We first use $\beta = h_2/h_3$ to remove h_2. Once $h_2 = 0$, h_1 must be non-zero (otherwise $h(x)$ would not be irreducible), so we can set $\alpha = \sqrt{h_1/h_3}$ and $\gamma = \sqrt{h_1^3/h_3}$ to obtain $h(x) = x^3 + x + h_0$.

We can then use δ to restrict f_6 to $\mathbb{F}_2$, then ε to force $f_5 = 0$, ϱ to force $f_4 = 0$ and finally ζ to have $f_3 = 0$. The curve equation is of the form

$$y^2 + (x^3 + x + h_0)y = f_7 x^7 + f_6 x^6 + f_2 x^2 + f_1 x + f_0, \tag{4.3}$$

where $f_6 \in \mathbb{F}_2$.

(IIa) $h_3 = 0$, $h_2 \neq 0$ and $h(x)$ is irreducible:

Using $\alpha = h_1/h_2$ and $\gamma = h_1^2/h_2$ we can force $h_2 = h_1 = 1$. Since $h(x)$ is irreducible, we have $\mathrm{TR}(h_1) = 1$ and we can use $\beta = \mathrm{HT}(h_1 + 1)$ to restrict $h(x)$ to $x^2 + x + 1$. The remaining freedom on β (i.e. $\beta \in \mathbb{F}_2$) allows us to impose $\mathrm{TR}(f_7) \cdot \mathrm{TR}(f_5) = 0$.

We can then use δ to force $f_6 = 0$, then ε to restrict f_4 to $\mathbb{F}_2$, then ϱ to force $f_3 = 0$ and finally ζ to have $f_2 = 0$. The curve equation is of the form

$$y^2 + (x^2 + x + 1)y = f_7 x^7 + f_5 x^5 + f_4 x^4 + f_1 x + f_0, \tag{4.4}$$

where $f_4 \in \mathbb{F}_2$ and $\mathrm{TR}(f_7) \cdot \mathrm{TR}(f_5) = 0$.

(III) $h_3 = h_2 = 0$ and $h_1 \neq 1$:

Taking $\alpha = (h_1^2/f_7)^{1/5}$ and $\gamma = (h_1^7/f_7)^{1/5}$, we can force both $h(x)$ and $f(x)$ to be monic. Once $h_1 = f_7 = 1$, we can use $\beta = h_0$ to obtain $h(x) = x$.

We can then use δ to force $f_6 = 0$, then ε to force $f_4 = 0$, ϱ to restrict f_2 to $\mathbb{F}_2$ and finally ζ to have $f_1 = 0$. The curve equation is of the form

$$y^2 + xy = x^7 + f_5x^5 + f_3x^3 + f_2x^2 + f_0, \tag{4.5}$$

where $f_2 \in \mathbb{F}_2$.

(IV) $h_3 = h_2 = h_1 = 0$ and $h_0 \neq 0$:

Taking $\alpha = (h_0^2/f_7)^{1/7}$ and $\gamma = h_0$, we can have $h(x) = 1$ and force $f(x)$ to be monic.

Once $f_7 = 1$, we can use $\beta = \sqrt{f_5}$ to remove f_5 from $f(x)$. We can then use δ to force $f_6 = 0$, then ε to force $f_4 = 0$, ϱ to force $f_2 = 0$ and finally ζ to restrict f_0 to $\mathbb{F}_2$. The curve equation is of the form

$$y^2 + y = x^7 + f_3x^3 + f_1x + f_0, \tag{4.6}$$

where $f_0 \in \mathbb{F}_2$.

Note that we did not include the non-singularity condition, nor conditions on the group order in the descriptions of the different types. In terms of isomorphism classes, Type Ia is the most common (with $\frac{2}{3}q^5 + O(q^4)$ classes), followed by Type IIa (with $\frac{3}{2}q^4 + O(q^3)$ classes), then Type III (with $2q^3 + O(q^2)$ classes) and finally Type IV (with $2q^2 + O(q)$ classes).

4.5. Type IV: $h(x) = 1$

In this section, we consider the optimal-performance case, namely Type IV. Curves of that type are preferred when computational speed is more important than flexibility in the choice of the curve (even then, there are enough isomorphism classes available for most applications).

Since the coefficients of the curve equation (the coefficients of $h(x)$ and $f(x)$) have a direct impact on the computations in Cantor's algorithm, it is quite natural to use isomorphisms to obtain an equivalent curve with "simpler" coefficients (i.e. getting coefficients equal to 0, restricting them to $\mathbb{F}_2$, etc). From the results of the previous section, we can assume that curves of Type IV are of the form

$$C : y^2 + y = x^7 + f_3x^3 + f_1x + f_0, \tag{4.7}$$

with $f_0 \in \mathbb{F}_2$.

4.5.1. Advantages of Type IV

As well as having all but two of the coefficients of the curve equation in $\mathbb{F}_2$ (and many of those being 0), these curves offer other advantages:

(1) The doubling is significantly faster than for other types of curves, and also much faster than the group addition.

(2) The curve C is not supersingular (see Theorem 1.2 in [SZ02] with $n = 3$). This is an important advantage over the genus-1 and genus-2 situation where curves with $h = c$ are supersingular if c is a constant. Hence our genus-3 curves are secure against specialised attacks as long as the order of the Picard group is divisible by a large prime.

(3) Since $h(x) = 1$, the 2-rank of the curve C is 0.

Given any non-zero element $\overline{D} = [u, v]$ of the Picard group in Mumford representation, its negative is

$$-\overline{D} = [u, -v - h \pmod{u}] = [u, v + 1]. \tag{4.8}$$

Since $v \not\equiv v + 1 \pmod{u}$ for all $\overline{D} \neq [1, 0]$, we have $\overline{D} \neq -\overline{D}$ and therefore $[2]\overline{D} \neq [1, 0]$ for all non-zero $\overline{D}$ in the Picard group of the curve, which shows that there are no non-trivial 2-torsion elements (and the group order is odd).

This simple fact is extremely useful for the halving formulas. It means that the doubling map is a one-to-one function, rather than a two-to-one as it is the case for the other curves considered in this chapter. The preimage of the doubling will be unique, removing the need for a potentially expensive verification step to find which of the preimage has odd order.

We will now give explicit formulas for the doubling and the halving of divisor classes, and cover both the most frequent case and all other possible special cases. Combined with the divisor class addition formulas in [ACD+05, ATW08, GKP04], this allows to program the most efficient implementation of genus-3 hyperelliptic curve group arithmetic.

4.5.2. Explicit doubling formulas

In the following, we give a complete study of all cases that can occur when performing doubling of a divisor class on a genus-3 hyperelliptic curve of Type IV, i.e. we assume that we are given a curve of the form (4.7) over a binary field. We consider the different cases by looking at the degree of the polynomial u_a in the Mumford representation of a divisor class $\overline{D}_a = [u_a, v_a]$, where u_a is monic, has degree at most 3 and v_a has smaller degree than u_a such that u_a divides $v_a^2 + v_a + f$.

We will give criteria to detect which case is present, depending on the coefficients of the polynomials u_a and v_a. Therefore, we follow the steps of Cantor's algorithm to see how the degrees of the polynomials behave during the doubling. The details of Cantor's algorithm for curves of genus 3 are given in Algorithm 22.

Algorithm 22 Cantor's doubling algorithm for genus-3 HEC in characteristic 2 with $h(x) = 1$ (Type IV)

INPUT: The divisor class $\overline{D} = [u_a, v_a]$
OUTPUT: The divisor class $\overline{E} = [u_c, v_c]$ such that $\overline{E} = [2]\overline{D}$

1: $u_1 \leftarrow u_a^2$, $v_1 \leftarrow v_a^2 + f \bmod u_1$
2: **if** $\deg(u_1) \leq 3$ **then**
3: $\quad u_c \leftarrow u_1$, $v_c \leftarrow v_1$
4: **else**
5: $\quad u_2 \leftarrow \text{monic}\left(\frac{f+v_1+v_1^2}{u_1}\right)$, $v_2 \leftarrow v_1 + 1 \bmod u_2$
6: $\quad$ **if** $\deg(u_2) \leq 3$ **then**
7: $\quad\quad u_c \leftarrow u_2$, $v_c \leftarrow v_2$
8: $\quad$ **else**
9: $\quad\quad u_c \leftarrow \text{monic}\left(\frac{f+v_2+v_2^2}{u_2}\right)$, $v_c \leftarrow v_2 + 1 \bmod u_c$
10: $\quad$ **end if**
11: **end if**
12: **return** $[u_c, v_c]$

Note that from now on, we will use the following notation: "Doubling $n \rightarrow m$" (shortened to DBLnm) stands for a doubling where the degree of the the first polynomial of the divisor class to be doubled is n and the degree of the first polynomial of the target divisor class (in Mumford representation) is m. We will use the same syntax for halving (shortened to HLVnm).

4.5.3. Distinguishing the cases

To distinguish the different doubling cases, we start with a divisor class $\overline{D}_a = [u_a, v_a]$ and investigate (depending on the degree of u_a) which degrees are possible for u_c in $\overline{D}_c = [u_c, v_c] = [2]\overline{D}_a$.

If u_a has degree 3, then the first step in Algorithm 22 computes $u_1 = u_a^2$ and $v_1 \equiv v_a^2 + f \bmod u_1$. We obtain

$$u_1 = u_a^2 = x^6 + u_{a2}^2 x^4 + u_{a1}^2 x^2 + u_{a0}^2,$$

and

$$\begin{aligned} v_1 &= v_a^2 + f(x) \pmod{u_1} \\ &= u_{a2}^2 x^5 + v_{a2}^2 x^4 + (u_{a1}^2 + f_3)x^3 + v_{a1}^2 x^2 \\ &\quad + (u_{a0}^2 + f_1)x + (f_0 + v_{a0}^2). \end{aligned} \tag{4.9}$$

Since u_1 has degree 6, we must do at least one reduction step, so we compute

$$u_2 = \text{Monic}\left(\frac{f + v_1 + v_1^2}{u_1}\right).$$

We now have different possibilities for $\deg(u_2)$, depending on the degree of v_1. Since $\deg(u_1) = 6$, the degree of v_1 is less than or equal to 5. We have the following three cases:

(1) When $\deg(v_1)$ is equal to 1, 2 or 3, the dominating part of the numerator comes from f. The degree of u_2 is then $\deg(u_2) = \deg(f) - \deg(u_1) = 1$. Cantor's algorithm will then output $u_c \leftarrow u_2$ of degree 1. This is case DBL31.

(2) When $\deg(v_1) = 4$, the numerator is dominated by v_1^2. The degree of u_2 is then $\deg(u_2) = \deg(v_1^2) - \deg(u_1) = 2$. Cantor's algorithm outputs $u_c \leftarrow u_2$ of degree 2. This is case DBL32.

(3) When $\deg(v_1) = 5$, the numerator is again dominated by v_1^2, but this time we have $\deg(u_2) = 4$. Note that we also have $\deg(v_2) \leq 3$. Cantor's algorithm will proceed with a second reduction step, computing u_c as

$$u_c = \text{Monic}\left(\frac{f + v_2 + v_2^2}{u_2}\right). \tag{4.10}$$

The numerator is once again dominated by f, and u_c has degree $\deg(f) - \deg(u_2) = 3$. This is case DBL33.

If u_a has degree 2, then $\deg(u_1) = 4$ and $\deg(v_1) \leq 3$. We must do one reduction step, with

$$u_2 = \text{Monic}\left(\frac{f + v_1 + v_1^2}{u_1}\right), \tag{4.11}$$

where the numerator is dominated by f. The degree of u_2 is $\deg(u_2) = \deg(f) - \deg(u_1) = 3$ and Cantor's algorithm will output $u_c \leftarrow u_2$ of degree 3. This is case DBL23.

Finally, if u_a has degree 1, then $\deg(u_1) = 2$ and $\deg(v_1) \leq 1$ and Cantor's algorithm outputs $u_c = u_1$ and $v_c = v_1$. This is case DBL12.

Doubling $3 \to 3$

This is in fact the most common case of doubling From the previous section, we know this will happen when $\deg(v_1) = 5$, which means $u_{a2}^2 \neq 0$, i.e. when $\deg(u_a) = 3$ and $u_{a2} \neq 0$.

We can now state the actual formula to double in the $3 \to 3$ case. This formula is taken from [FWW06, Table XXVI], although we adapted the notation to the one used in this chapter.

Algorithm 23 (DBL33, $h(x) = 1$, $f(x) = x^7 + f_3x^3 + f_1x + f_0$)

INPUT: $\overline{D} = [x^3 + u_{a2}x^2 + u_{a1}x + u_{a0}, v_{a2}x^2 + v_{a1}x + v_{a0}]$, $u_{a2} \neq 0$
OUTPUT: $[2]\overline{D} = [x^3 + u_{c2}x^2 + u_{c1}x + u_{c0}, v_{c2}x^2 + v_{c1}x + v_{c0}]$

1: $s_0 \leftarrow u_{a2}^2$, $s_1 \leftarrow u_{a1}^2$, $s_2 \leftarrow v_{a2}^2$, $s_3 \leftarrow f_3 + s_1$, $s_4 \leftarrow s_0^{-1}$ ▷ 1I+3S
2: $s_5 \leftarrow s_4s_2$, $s_6 \leftarrow s_4s_3$, $u_{c1} \leftarrow s_5^2 + s_0$, $s_7 \leftarrow s_0u_{c1}$ ▷ 3M+1S
3: $s_8 \leftarrow s_6^2 + s_1 + s_7$, $s_9 \leftarrow s_7 + s_3$, $s_{10} \leftarrow s_2u_{c1}$, $s_{11} \leftarrow s_2s_8$ ▷ 2M+1S
4: $s_{12} \leftarrow s_{11} + f_0 + v_{a0}^2$, $s_{13} \leftarrow s_{10} + v_{a1}^2 + s_4$, $s_{14} \leftarrow s_4^2$ ▷ 3S
5: $s_{15} \leftarrow (s_0 + s_2)(s_8 + s_{14}) + s_{11} + f_1 + u_{a0}^2 + s_4$, $u_{c2} \leftarrow s_9^2$ ▷ 1M+2S
6: $u_{c0} \leftarrow u_{c2}u_{c1} + s_{13}^2 + s_{14}$, $s_{16} \leftarrow s_9u_{c0}$, $s_{17} \leftarrow s_9u_{c1}$ ▷ 3M+1S
7: $s_{18} \leftarrow s_9u_{c2}$, $v_{c2} \leftarrow s_{13} + s_{18}$, $v_{c1} \leftarrow s_{15} + s_{17}$, $v_{c0} \leftarrow s_{12} + s_{16}$ ▷ 1M
8: **return** $[x^3 + u_{c2}x^2 + u_{c1}x + u_{c0}, v_{c2}x^2 + v_{c1}x + v_{c0}]$ ▷ 1I+10M+11S

Doubling $3 \to 2$

From Subsection 4.5.3 (2), we know that this case occurs when $\deg(v_1) = 4$. And from (4.9) this happens if and only if $\deg(u_a) = 3$, $u_{a2} = 0$ and $v_{a2} \neq 0$.

Algorithm 24 (DBL32, $h(x) = 1$, $f(x) = x^7 + f_3x^3 + f_1x + f_0$)

INPUT: $\overline{D} = [x^3 + u_{a1}x + u_{a0}, v_{a2}x^2 + v_{a1}x + v_{a0}]$, $v_{a2} \neq 0$
OUTPUT: $[2]\overline{D} = [x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$

1: $s_0 \leftarrow v_{a2}^2$, $s_1 \leftarrow s_0^{-1}$, $u_{c1} \leftarrow s_1^2$, $s_2 \leftarrow f_3 + u_{a1}^2$, $s_3 \leftarrow s_2^2$ ▷ 1I+4S
2: $u_{c0} \leftarrow u_{c1}s_3$, $s_4 \leftarrow s_1 + s_2$, $s_5 \leftarrow s_4u_{c0}$ ▷ 2M
3: $s_6 \leftarrow v_{a1}^2 + (s_0 + s_4)(u_{c0} + u_{c1}) + s_5 + s_1$, $s_7 \leftarrow s_6u_{c0}$ ▷ 2M+1S
4: $s_8 \leftarrow s_6u_{c1}$, $v_{c1} \leftarrow f_1 + u_{a0}^2 + s_5 + s_8$, $v_{c0} \leftarrow f_0 + 1 + v_{a0}^2 + s_7$ ▷ 1M+2S
5: **return** $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$ ▷ 1I+5M+7S

Doubling $3 \to 1$

From Subsection 4.5.3 (1), we see that this case occurs when $\deg(v_1)$ is less than or equal to 3, and from (4.9) this happens when $\deg(u_a) = 3$, $u_{a2} = 0$ and $v_{a2} = 0$.

Since any divisor class must satisfy $u_a \mid v_a^2 + hv_a - f$, it is easy to show that we must also have $v_{a1} = 0$. We can therefore assume that the input divisor class has the form $[x^3 + u_{a1}x + u_{a0}, v_{a0}]$ and the output divisor class is of the form $[x + u_{c0}, v_{c0}]$.

Note that in the operation count we assumed f_3^2 was precomputed. However, because of the low probability of having to perform the $3 \to 1$ doubling, it may be considered more useful (in terms of memory usage) to compute f_3^2 only when it is needed, giving an operation count of 2M+6S.

Algorithm 25 (DBL31, $h(x) = 1$, $f(x) = x^7 + f_3x^3 + f_1x + f_0$)

INPUT: $\overline{D} = [x^3 + u_{a1}x + u_{a0},\ v_{a0}]$

OUTPUT: $[2]\overline{D} = [x + u_{c0},\ v_{c0}]$

1: $s_0 \leftarrow u_{a1}^2,\ s_1 \leftarrow f_3^2,\ s_2 \leftarrow s_0^2,\ u_{c0} \leftarrow s_1 + s_2,\ s_4 \leftarrow u_{c0}^2$ ▷ 3S

2: $v_{c0} \leftarrow u_{c0}\left((s_0 + f_3)s_4 + (u_{a0}^2 + f_1)\right) + v_{a0}^2 + f_0 + 1$ ▷ 2M+2S

3: **return** $[x + u_{c0},\ v_{c0}]$ ▷ 2M+5S

Doubling $2 \to 3$

As stated in Section 4.5.3, this is the only case that can occur when $\deg(u_a) = 2$. The first step of Cantor's algorithm gives us $u_1 = u_a^2 = x^4 + u_{a1}^2x^2 + u_{a0}^2$ and

$$v_1 = (f_3 + u_{a0}^2 + u_{a1}^4)x^3 + v_{a1}^2x^2 + (u_{a1}^2u_{a0}^2 + f_1)x + (f_0v_{a0}^2), \tag{4.12}$$

after which one reduction step is performed to obtain u_c and v_c. The formula is as follows:

Algorithm 26 (DBL23, $h(x) = 1$, $f(x) = x^7 + f_3x^3 + f_1x + f_0$)

INPUT: $\overline{D} = [x^2 + u_{a1}x + u_{a0}, v_{a1}x + v_{a0}]$

OUTPUT: $[2]\overline{D} = [x^3 + u_{c2}x^2 + u_{c1}x + u_{c0}, v_{c2}x^2 + v_{c1}x + v_{c0}]$

1: $u_{c1} \leftarrow u_{a1}^2,\ s_0 \leftarrow f_3 + u_{c1}^2,\ s_1 \leftarrow s_0 + u_{a0}^2,\ u_{c2} \leftarrow s_1^2,\ s_2 \leftarrow v_{a1}^2$ ▷ 5S

2: $s_3 \leftarrow s_2 + s_1u_{a1},\ u_{c0} \leftarrow s_3^2,\ s_4 \leftarrow s_1u_{c0},\ s_5 \leftarrow s_1u_{c2}$ ▷ 3M+1S

3: $v_{c2} \leftarrow s_2 + s_5,\ v_{c1} \leftarrow f_1 + s_0u_{c1},\ v_{c0} \leftarrow f_0 + v_{a0}^2 + 1 + s_4$ ▷ 1M+1S

4: **return** $[x^3 + u_{c2}x^2 + u_{c1}x + u_{c0},\ v_{c2}x^2 + v_{c1}x + v_{c0}]$ ▷ 4M+7S

Doubling $1 \rightarrow 2$

This is the last case which can occur when performing a doubling. Since $\deg(u_a^2) = 2 < 3$, only the first step of Cantor's algorithm is necessary, and we obtain $u_c = u_1 = u_a^2 = x^2 + u_{a0}^2$ and

$$v_c = v_1 = (f_1 + u_{a0}^6 + f_3 u_{a0}^2)x + (v_{a0}^2 + f_0). \tag{4.13}$$

We get the very short formula:

Algorithm 27 (DBL12, $h(x) = 1$, $f(x) = x^7 + f_3x^3 + f_1x + f_0$)

INPUT: $\overline{D} = [x + u_{a0},\ v_{a0}]$
OUTPUT: $[2]\overline{D} = [x^2 + u_{c0},\ v_{c1}x + v_{c0}]$

1: $u_{c0} \leftarrow u_{a0}^2$, $v_{c0} \leftarrow f_0 + v_{a0}^2$, $v_{c1} \leftarrow f_1 + u_{c0}(f_3 + u_{c0}^2)$ ▷ 1M+3S
2: **return** $[x^2 + u_{c0},\ v_{c1}x + v_{c0}]$ ▷ 1M+3S

4.5.4. Explicit halving formulas

Having developed formulas for all the possible cases of divisor class doubling, we can now look at halving these same classes of our proposed genus-3 curves over binary fields. Our general approach will consist in inverting (or "backtracking") each one of the doubling cases to obtain the halving formulas. We will therefore have five cases of halvings:

- Halving $3 \rightarrow 3$ (from the doubling $3 \rightarrow 3$)
- Halving $2 \rightarrow 3$ (from the doubling $3 \rightarrow 2$)
- Halving $1 \rightarrow 3$ (from the doubling $3 \rightarrow 1$)
- Halving $3 \rightarrow 2$ (from the doubling $2 \rightarrow 3$)
- Halving $2 \rightarrow 1$ (from the doubling $1 \rightarrow 2$)

Before going into the specifics of each formula, let us consider how to distinguish between the different cases. Let us consider the halving of a divisor class $[u_c, v_c]$ known to come from the doubling of a divisor class $[u_a, v_a]$ (to be determined later):

- If $\deg(u_c) = 1$, then we can only be in the $1 \rightarrow 3$ case.

- If $\deg(u_c) = 2$, then $\deg(u_a)$ was either 1 (doubling $1 \to 2$)—in which case $u_c(x)$ is of the form $x^2 + u_{c0}$—or 3 (doubling $3 \to 2$). To have a simple distinguishing condition, we would like to say that if $[u_c, v_c]$ comes from a doubling $3 \to 2$ then $u_c(x)$ is of the form $x^2 + u_{c1}x + u_{c0}$ with $u_{c1} \neq 0$, and indeed, an easy computation from the doubling formula shows that $u_{c1} = 1/v_{a2}^4$, where $v_{a2} \neq 0$ as we are coming from the $3 \to 2$ doubling case.
- If $\deg(u_c) = 3$, then $\deg(u_a)$ was either 2 (doubling $2 \to 3$) or 3 (doubling $3 \to 3$). There is no direct way to distinguishing between these two cases simply by looking at the form of u_c and v_c. However, the doubling formulas do present us with a natural candidate when we notice that the $3 \to 3$ doubling contains an inversion while the $2 \to 3$ doubling does not. Not surprisingly, the same situation happens in the halving formulas. If we assume that $[u_c, v_c]$ is in the halving $3 \to 3$ case and try to work backwards through the $3 \to 3$ doubling, we need to compute the inverse of $u_{c0} + v_{c1}^2 + u_{c2}(u_{c1} + u_{c2}^2)$ (or it's square root), so the operation cannot be valid if this value is 0 (i.e. it must be $\neq 0$). On the other hand, if we take the result of a $2 \to 3$ doubling and substitute the values of the u_{c0}, u_{c1}, u_{c2} and v_{c1} (in terms of the coefficients of u_a and v_a) in the expression $u_{c0} + v_{c1}^2 + u_{c2}(u_{c1} + u_{c2}^2)$, then we can verify that it must always be 0. We can therefore use the value of $u_{c0} + v_{c1}^2 + u_{c2}(u_{c1} + u_{c2}^2)$ to safely distinguish between the two cases.

Now that the different cases can be identified, we can look at the formulas. To have a more "standard" look, they are written with input $[u_a, v_a]$ and output $[u_c, v_c] = [\frac{1}{2}][u_a, v_a]$, so the condition to distinguish between the cases are u_{a1} equal or not to 0 when $\deg(u_a) = 2$ and $u_{a0} + v_{a1}^2 + u_{a2}(u_{a1} + u_{a2}^2)$ equal or not to 0 when $\deg(u_a) = 3$.

Halving $3 \to 3$

We can now optimise the halving $3 \to 3$. In general, we cannot distinguish this case from the halving $3 \to 2$ until $s_4 = u_{a0} + v_{a1}^2 + u_{a2}(u_{a1} + u_{a2}^2)$ has been computed. If $s_4 = 0$, then we must change to the same line of Algorithm 11. Note that both $u_{a0}\sqrt{u_{a2}}$ and $u_{a2}\sqrt{u_{a2}}$ are needed in both the $3 \to 3$ and $3 \to 2$ halvings, so they can be computed before we distinguish the two cases.

Algorithm 28 (HLV33, $h(x) = 1$, $f(x) = x^7 + f_3x^3 + f_1x + f_0$)

INPUT: $\overline{D} = [x^3 + u_{a2}x^2 + u_{a1}x + u_{a0}, v_{a2}x^2 + v_{a1}x + v_{a0}]$

OUTPUT: $[\frac{1}{2}]\overline{D} = [x^3 + u_{c2}x^2 + u_{c1}x + u_{c0}, v_{c2}x^2 + v_{c1}x + v_{c0}]$

1: $s_0 \leftarrow \sqrt{u_{a2}}$, $s_1 \leftarrow u_{a2}s_0$, $s_2 \leftarrow u_{a0}s_0$, $s_3 \leftarrow v_{a2} + s_1$ ▷ 2M+1SR

2: $s_4 \leftarrow u_{a0} + s_3^2 + u_{a1}u_{a2}$, $s_5 \leftarrow \sqrt{s_4}$, $s_6 \leftarrow s_5^{-1}$ ▷ 1I+1M+1S+1SR

3: $s_7 \leftarrow u_{a1}s_6$, $s_8 \leftarrow s_0 + s_7$, $s_9 \leftarrow s_6\sqrt{s_6 + u_{a1}}$ ▷ 2M+1SR
4: $s_{10} \leftarrow s_3 + u_{a1}s_9 + s_5$, $s_{11} \leftarrow s_8 + f_3$, $s_{12} \leftarrow s_{11} + s_4s_8^2 + s_7$ ▷ 2M+1S
5: $s_{13} \leftarrow s_{12}s_9$, $s_{14} \leftarrow v_{a1} + u_{a1}s_0 + (s_4 + s_{12})(s_9 + s_6) + s_{13} + s_5$ ▷ 3M
6: $s_{15} \leftarrow v_{a0} + s_2 + s_{13}$, $v_{c2} \leftarrow \sqrt{s_9}$, $v_{c1} \leftarrow \sqrt{s_{10}}$ ▷ 2SR
7: $v_{c0} \leftarrow \sqrt{s_{15} + f_0}$, $u_{c2} \leftarrow \sqrt{s_6}$, $u_{c1} \leftarrow \sqrt{s_{11}}$, $u_{c0} \leftarrow \sqrt{s_{14} + f_1}$ ▷ 4SR
8: **return** $[x^2 + u_{c0},\ v_{c1}x + v_{c0}]$ ▷ 1I+10M+2S+9SR

Halving $2 \to 3$

Since this case of the halving is the inverse of a $3 \to 2$ doubling, we know that the output must be of the form $[x^3 + u_{c1}x + u_{c0},\ v_{c2}x^2 + v_{c1}x + v_{c0}]$. However, the output has one more coefficient than the input, and it is not enough to simply reverse the doubling formula—doing so would leave us with q possible choices for the output, which is clearly impossible as the halving operation is injective.

To solve this problem, we must recall the last condition in Theorem 2.16, i.e. that u_c must divide $v_c^2+v_c+f$ if $[u_c, v_c]$ is a divisor class. Computing the coefficient of x^2 in

$$(v_{c2}x^2 + v_{c1}x + v_{c0})^2 + (v_{c2}x^2 + v_{c1}x + v_{c0}) + f \pmod{x^3 + u_{c1}x + u_{c0}},$$

we find that $v_{c2} + v_{c1}^2 + u_{c1}v_{c2}^2$ must be 0 (since the whole equation must equal 0), giving us the relation $v_{c1} = \sqrt{v_{c2} + u_{c1}v_{c2}^2}$ which allows us to complete the formula.

Algorithm 29 (HLV23, $h(x) = 1$, $f(x) = x^7 + f_3x^3 + f_1x + f_0$)

INPUT: $\overline{D} = [x^2 + u_{a1}x + u_{a0}, v_{a1}x + v_{a0}]$
OUTPUT: $[\frac{1}{2}]\overline{D} = [x^3 + u_{c1}x + u_{c0}, v_{c2}x^2 + v_{c1}x + v_{c0}]$

1: $s_0 \leftarrow \sqrt{u_{a1}}$, $s_1 \leftarrow s_0^{-1}$, $v_{c2} \leftarrow \sqrt{s_1}$, $s_2 \leftarrow \sqrt{u_{a0}}$, $s_3 \leftarrow s_1s_2$ ▷ 1I+1M+3SR
2: $s_6 \leftarrow s_1u_{a0}$, $u_{c1} \leftarrow \sqrt{s_3 + f_3}$, $s_4 \leftarrow v_{c2} + u_{c1}s_1$, $v_{c1} \leftarrow \sqrt{s_4}$ ▷ 2M+2SR
3: , $s_5 \leftarrow s_3 + s_0$, $s_7 \leftarrow (s_4 + s_6)u_{a1}$, $s_8 \leftarrow u_{a0} + u_{a1}^2$ ▷ 1M+1S
4: $s_9 \leftarrow s_5s_8$, $s_{10} \leftarrow s_5u_{a1}$, $u_{c0} \leftarrow \sqrt{f_1 + s_9 + s_7 + v_{a1}}$ ▷ 2M+1SR
5: $s_{11} \leftarrow s_4 + s_6 + s_{10}$, $v_{c0} \leftarrow \sqrt{f_0 + 1 + s_{11}u_{a0} + v_{a0}}$ ▷ 1M+1SR
6: **return** $[x^3 + u_{c1}x + u_{c0},\ v_{c2}x^2 + v_{c1}x + v_{c0}]$ ▷ 1I+7M+1S+7SR

Halving $1 \to 3$

Just as in the $2 \to 3$ case, the output has more coefficients than the input, giving us difficulties to reverse the doubling formula. This time, the output must be of the form $[x^3 + u_{c1}x + u_{c0},\, v_{c0}]$, and once again the solution can be found in Theorem 2.16. We compute the coefficient of x in

$$v_{c0}^2 + v_{c0} + f(x) \pmod{x^3 + u_{c1}x + u_{c0}}$$

to find that $f_1 + u_{c0}^2 + u_{c1}f_3 + u_{c1}^3$ must be 0, and we can complete the formula using the relation $u_{c0} = \sqrt{f_1 + u_{c1}(f_3 + u_{c1}^2)}$.

Algorithm 30 (HLV13, $h(x) = 1$, $f(x) = x^7 + f_3x^3 + f_1x + f_0$)

INPUT: $\overline{D} = [x + u_{a0}, v_{a0}]$

OUTPUT: $[\frac{1}{2}]\overline{D} = [x^3 + u_{c1}x + u_{c0},\, v_{c0}]$

1: $s_0 \leftarrow \sqrt{u_{a0}} + f_3,\ u_{c1} \leftarrow \sqrt{s_0},\ s_1 \leftarrow u_{a0}^2$ ▷ 1S+2SR

2: $s_2 \leftarrow (f_3 + s_0)u_{c1},\ s_3 \leftarrow (f_3 + s_0)s_1,\ u_{c0} \leftarrow \sqrt{f_1 + s_2}$ ▷ 2M+1SR

3: $v_{c0} \leftarrow \sqrt{v_{a0} + u_{a0}\,(s_3 + s_2) + f_0 + 1}$ ▷ 1M+1SR

4: **return** $[x^3 + u_{c1}x + u_{c0},\, v_{c2}x^2 + v_{c1}x + v_{c0}]$ ▷ 3M+1S+4SR

Halving $3 \to 2$

Although reversing the $2 \to 3$ doubling formula can be done in 2M and 5SR, distinguishing the $3 \to 2$ halving from the $3 \to 3$ case requires a few more operations.

Algorithm 31 (HLV32, $h(x) = 1$, $f(x) = x^7 + f_3x^3 + f_1x + f_0$)

INPUT: $\overline{D} = [x^3 + u_{a2}x^2 + u_{a1}x + u_{a0}, v_{a2}x^2 + v_{a1}x + v_{a0}]$

OUTPUT: $[\frac{1}{2}]\overline{D} = [x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$

1: $s_0 \leftarrow \sqrt{u_{a2}},\ s_1 \leftarrow u_{a2}s_0,\ s_2 \leftarrow u_{a0}s_0,\ s_3 \leftarrow v_{a2} + s_1$ ▷ 2M+1SR

2: $s_4 \leftarrow u_{a0} + s_3^2 + u_{a1}u_{a2},\ u_{c1} \leftarrow \sqrt{u_{a1}},\ u_{c0} \leftarrow u_{a1} + \sqrt{s_0 + f_3}$ ▷ 1M+1S+2SR

3: $v_{c1} \leftarrow \sqrt{v_2 + s_1},\ v_{c0} \leftarrow \sqrt{v_{a0} + f_0 + 1 + s_2}$ ▷ 2SR

4: **return** $[x^2 + u_{c1}x + u_{c0},\, v_{c1}x + v_{c0}]$ ▷ 3M+1S+5SR

Halving $2 \to 1$

This is the final case of halving, and the simplest one.

Algorithm 32 (HLV21, $h(x) = 1$, $f(x) = x^7 + f_3x^3 + f_1x + f_0$)

INPUT: $\overline{D} = [x^2 + u_{a0}, v_{a1}x + v_{a0}]$
OUTPUT: $[\frac{1}{2}]\overline{D} = [x + u_{c0}, v_{c0}]$

1: $u_{c0} \leftarrow \sqrt{u_{a0}}$, $v_{c0} \leftarrow \sqrt{v_{a0} + f_0}$
2: **return** $[x + u_{c0}, v_{c0}]$ ▷ 2SR

4.6. Halving for other types of curves

In this section, we consider halving formulas for curves of genus 3 with h irreducible (and non-constant), i.e. curves of Type Ia, IIa, and III.

In general, it would be safe to say that the cost of the halving operation increases with the degree of h, but this is in a way offset by having a larger number of isomorphism classes (in particular when h is irreducible), giving us more flexibility in the choice of the curves.

Unlike the curves in the previous section (Type IV), the doubling of a divisor class admits two preimages, and we must distinguish which of the two has odd order. Since the doubling is a 2-to-1 map, the structure of the special cases will be somewhat altered. How to deal with this altered situation will be described in Subsection 4.6.1.

In the following subsections, we will study the three types of curves in increasing order of complexity (i.e. increasing the degree of h). For each type of curve, we will "define" the different cases (i.e. describe how to distinguish them) and obtain necessary and sufficient conditions under which a divisor class can be halved. This will allow us to give a simple criterion for the curve to have order $2r$, where r is odd, after which we give the explicit formulas for the most common case.

Finally, we will analyse the results in Subsection 4.6.5.

4.6.1. Halving $3 \to 3$ versus special cases

If we look at the doubling algorithm when $\deg(h) > 0$, the most obvious difference is that we cannot ignore the gcd of h and u_a. If $\gcd(u_a, h) \neq 1$, we first divide u_a by $\gcd(u_a, h)$, and reduce v_a accordingly, after which the "normal" structure of special cases applies (clearly only the doublings $2 \to 3$ and $1 \to 2$ are possible if $u_a / \gcd(u_a, h)$ is different from 1).

The observation on $\gcd(u_a, h) \neq 1$ is very indicative of the problem we face with the special cases of halving, but also hints at the solution. In the curves

we are interested in this section, the doubling is a 2-to-1 function, so to compute the halving we will find two possible preimages, but these preimages could have different degrees (which complicates the distinction between the different special cases). On the other hand, the difference between the two preimages is always the unique divisor class of order 2, so once we can compute a preimage the other one can be found using Cantor's algorithm (adding the divisor class of order 2). Note that the divisor class of order 2 is of the form $[h, v_h]$ when h is irreducible, and of the form $[x, \sqrt{f_0}]$ when h is a square or a cube.

To denote the halving cases, we will base ourselves on the lowest degree of the preimage, and then aggregate the degree of the other preimage if it is different. For example, HLV32/33 indicates that the input has degree 3, that one of the two preimages has degree 2 and the second one has degree 3. If both preimages have the same degree, we keep the same notation as before (for example HLV23). The main advantage of this notation is that the preimage of lowest degree is generally the one that closely matches the corresponding case for Type-IV curves.

In fact, when the preimages have distinct degrees, the second preimage can often be found simply by adding the (unique) divisor class of order 2 to the first preimage using Cantor's algorithm without the reduction step (as long as the total degree remains less than 3), and it is usually more efficient to compute it explicitly in this way. When a reduction step would be required to add the divisor class of order 2, we could still use Cantor's algorithm, but it appears more efficient to go back to inverting the doubling. We observe that those cases are due to certain coefficients being 0 in the doubling, leading to "degenerate" quadratic equations, for example $z^2 + 0z = \alpha$ (which has a double root instead of two distinct ones).

4.6.2. Type III: $h(x) = x$

According to Section 4.4, curves of Type III are of the form

$$C : y^2 + xy = x^7 + f_5x^5 + f_3x^3 + f_2x^2 + f_0, \tag{4.14}$$

where $f_2 \in \mathbb{F}_2$. The Picard group of these curves has precisely one divisor class of order 2, which is of the form $[x, \sqrt{f_0}]$.

Theorem 4.2. Let $\overline{D}_a = [u_a, v_a]$ be a divisor class in $\mathrm{Pic}^0(C)$. If $\deg(u_a) = 3$, then $\overline{D}_a$ can be halved if and only if $\mathrm{TR}(u_{a1}((u_{a2}^2 + f_5)u_{a2} + v_{a2}^2 + u_{a0})) = 0$. If $\deg(u_a) = 2$, then $\overline{D}_a$ can be halved if and only if $\mathrm{TR}(u_{a1}((u_{a0} + u_{a1}^2)(u_{a0} + f_5) + u_{a1}^4 + f_3)) = 0$. If $\deg(u_a) = 1$, then $\overline{D}_a$ can be halved if and only if $\mathrm{TR}(u_{a0}(u_{a0}^2(u_{a0}^2 + f_5) + f_3) + f_2) = 0$.

Proof. To halve a divisor class $\overline{D}_a = [u_a, v_a]$, we assume that it is the image under the doubling of a divisor class $\overline{D}_c = [u_c, v_c]$. To perform the halving, we work our way backwards through the doubling of $\overline{D}_c$, trying to solve for the coefficients of u_c and v_c given the coefficients of u_a and v_a (the form of u_c and v_c are determined by the halving case).

In cases HLV33, HLV23 and HLV13, the halving requires us to solve an equation of the form $z^2 + z + \alpha = 0$ at some point in the computations. If $\overline{D}_a$ is indeed equal to $[2]\overline{D}_c$ for some $\mathbb{F}_{2^d}$-rational divisor class $\overline{D}_c$, then an $\mathbb{F}_{2^d}$-rational root of $z^2 + z + \alpha = 0$ must exist (since all the operations in Cantor's algorithm are performed over $\mathbb{F}_{2^d}$). If $\mathrm{TR}(\alpha) = 1$, then no such root can exist, so a divisor class must have $\mathrm{TR}(\alpha) = 0$ if we want to halve it.

For cases HLV21/22 and HLV32/33, it is always possible to halve them, but there are special conditions on the coefficients of u_a and v_a and it can be shown that these conditions force $\alpha = 0$ (and obviously $\mathrm{TR}(\alpha) = 0$). This gives us the necessity of the trace conditions.

To complete the proof, we must show that the trace conditions are also sufficient. To do this, we show that as long as that condition holds for a reduced divisor, then applying one of the halving formula to this divisor will return an output that is a valid divisor. By construction (of the formula), the double of that new divisor must be the input of our halving, hence this input can be halved. Note that being able to compute two $\mathbb{F}_{2^d}$-rational polynomials u_c and v_c with the halving formulas is not sufficient on its own to give us a divisor. We must also verify that $v_c^2 + v_c h + f$ is divisible by u_c.

We explain how to do this in the HLV33 case, the other cases follow the same pattern. We begin with a divisor class $[u_a, v_a]$, i.e. $v_a^2 + v_a h + f \equiv 0 \pmod{u_a}$. The coefficients of x^0, x^1 and x^2 in this equality give us 3 coefficient identities, the "divisibility conditions". To obtain the halving formulas, we compute a sequence of pairs of polynomials $[u_i, v_i]$ which should all be semi-reduced divisors if we want the output to be a reduced divisor (rather than a random pair of polynomials).

From $[u_a, v_a]$, we first compute $[u_2, v_2]$ using the polynomial equations

$$\begin{aligned} u_a &= \mathrm{Monic}\left(\frac{v_2^2 + v_2 h + f}{u_2}\right), \\ v_a &\equiv v_2 + h(x) \pmod{u_a}, \\ v_2^2 + v_2 h + f &\equiv 0 \pmod{u_2} \end{aligned}$$

(working backwards through the second reduction and making sure we have a semi-reduced divisor). These equations give us 10 identities that must be satisfied by the coefficients of u_2 and v_2. We use 7 of these identities to compute the coefficients, and the 3 remaining identities become our new divisibility conditions. To show that $[u_c, v_c]$ is a semi-reduced divisor, we use the 7 identities of the halving formula to show that the 3 divisibility conditions of $[u_c, v_c]$ imply the 3 new divisibility conditions (once all 10 identities are satisfied, so are the 3 polynomial equations).

We then repeat the same idea to show that $[u_1, v_1]$ (first reduction) is also a semi-reduced divisor: To compute the coefficients of u_1 and v_1, we used 9 of the

13 coefficient identities in the equations

$$\begin{aligned} u_2 &= \text{Monic}\left(\frac{v_1^2 + v_1 h + f}{u_1}\right), \\ v_2 &\equiv v_1 + h \pmod{u_1}, \\ v_1^2 + v_1 h + f &\equiv 0 \pmod{u_1}. \end{aligned}$$

We are left with 4 divisibility conditions, which can be shown to be implied by the 3 divisibility conditions on u_1 and v_1 (once again using the 9 identities of the halving formula to perform the simplifications).

To finish, we have to show that $v_0^2 + v_0 h + f \equiv 0 \pmod{u_0}$, where $u_0 = \sqrt{u_1}$ and $v_0 \equiv v_1 \pmod{u_0}$ (i.e. performing the composition step backwards). This comes directly from $v_1^2 + v_1 h + f \equiv 0 \pmod{u_1}$. Since this halving case comes from $\gcd(u_c, h) = 1$ in the doubling of $[u_c, v_c]$ (the preimage of the doubling), we have $u_c = u_0$ and $v_c = v_0$ and all the divisibility conditions are already obtained.

To complete the proof, this process is repeated for the other halving cases, showing that in all cases the preimages computed are valid divisors if the trace conditions are satisfied. Note that for the HLV21/22 and HLV32/33 cases there is only one possible choice for u_0 and v_0. The first preimage (of lower degree) corresponds to $\gcd(u_c, h) = 1$ and no further work is required. The second preimage corresponds to $\gcd(u_c, h) = x$ and the divisibility conditions come from the addition of the reduced divisor $[u_0, v_0]$ to the reduced divisor of order 2 (using Cantor's algorithm, which does not require any reduction step in this case). □

Corollary 4.3. The Picard group of the curve C given by (4.14) has order $2r$, where r is odd, if and only if $f_2 = 1$.

Proof. The Picard group of the curve C has exactly one divisor class of order 2, namely $[x, \sqrt{f_0}]$. The order of the Picard group is divisible by 4 if and only if $[x, \sqrt{f_0}]$ can be halved. From Theorem 4.2, this is possible if and only if $\text{TR}(f_2) = 0$. Since $f_2 \in \mathbb{F}_2$, we find that $\text{Pic}^0(C)$ has a divisor class of order 4 if and only if $f_2 = 0$. □

Observe that if $\overline{D}_{c_1}$ and $\overline{D}_{c_2}$ are the two preimages of $\overline{D}_a$ under the doubling, then $\overline{D}_{c_1} - \overline{D}_{c_2} = [x, \sqrt{f_0}] = \overline{D}_{c_2} - \overline{D}_{c_1}$, i.e. the difference of two preimages is the unique divisor class of order 2. This observation allows us to distinguish the different special cases.

Remark 4.4. Let $\overline{D}_a = [u_a, v_a]$ be a divisor class in $\text{Pic}^0(C)$ that can be halved and $\overline{D}_c = [u_c, v_c] = [\frac{1}{2}]\overline{D}_a$ its preimage (under the doubling) of odd order.

(1a) If $\deg(u_a) = 3$ and $v_{a2}^2 + u_{a2}(u_{a2}^2 + u_{a1} + f_5) + \sqrt{u_{a2}} + u_{a0} \neq 0$, then $\deg(u_c) = 3$ and we are in case HLV33.

(1b) If $\deg(u_a) = 3$ and $v_{a2}^2 + u_{a2}(u_{a2}^2 + u_{a1} + f_5) + \sqrt{u_{a2}} + u_{a0} = 0$, then $\deg(u_c) = 2$ or 3 (with $u_{c0} = 0$ in the second case) and we are in case HLV32 or HLV33.

(2a) If $\deg(u_a) = 2$ and $u_{a1} \neq 0$, then $\deg(u_c) = 2$ and we are in case HLV23.

(2b) If $\deg(u_a) = 2$ and $u_{a1} = 0$, then $\deg(u_c) = 1$ or 2 (with $u_{c0} = 0$ in the second case) and we are in case HLV21 or HLV22.

(3) If $\deg(u_a) = 1$, then $\deg(u_c) = 3$ and we are in case HLV13.

We obtain the following halving formulas:

Algorithm 33 (HLV33, $h(x) = x$, $f(x) = x^7 + f_5x^5 + f_3x^3 + x^2 + f_0$)

INPUT: $\overline{D} = [x^3 + u_{a2}x^2 + u_{a1}x + u_{a0}, v_{a2}x^2 + v_{a1}x + v_{a0}]$

OUTPUT: $[\frac{1}{2}]\overline{D} = [x^3 + u_{c2}x^2 + u_{c1}x + u_{c0}, v_{c2}x^2 + v_{c1}x + v_{c0}]$

1: $s_0 \leftarrow \sqrt{u_{a2}}$, $s_1 \leftarrow u_{a1} + f_5$, $s_2 \leftarrow s_0u_{a2} + v_{a2}$, $s_3 \leftarrow s_1u_{a2}$ ▷ 2M+1SR

2: $s_4 \leftarrow s_0 + s_2^2 + s_3 + u_{a0}$, $s_5 \leftarrow s_4u_{a1}$, $s_6 \leftarrow s_0u_{a1} + v_{a1} + 1$ ▷ 2M+1S

3: $s_7 \leftarrow s_0u_{a0} + v_{a0}$, $s_8 \leftarrow s_2 + f_3 + (s_4 + s_1)(u_{a2} + u_{a1}) + s_3 + s_5$ ▷ 2M

4: $s_9 \leftarrow s_4^{-1}$, $s_{10} \leftarrow \sqrt{s_9}$, $s_{11} \leftarrow s_{10}s_1$, $s_{12} \leftarrow s_{11} + s_0$ ▷ 1I+1M+1SR

5: $s_{13} \leftarrow \mathrm{HT}(s_5)$, $s_{14} \leftarrow s_9s_{13}$ ▷ 1M+1HT

6: $s_{15} \leftarrow s_2 + (s_{14} + s_{10})(s_4 + s_1) + s_{11} + s_{13}$, $s_{16} \leftarrow s_{14}s_8 + s_7$ ▷ 2M

7: $s_{17} \leftarrow s_{14} + f_5$, $u_{c2} \leftarrow \sqrt{s_{17}}$, $s_{18} \leftarrow s_{15} + f_3$, $u_{c1} \leftarrow \sqrt{s_{18}}$ ▷ 2SR

8: $s_{19} \leftarrow s_{16} + f_1$, $u_{c0} \leftarrow \sqrt{s_{19}}$, $s_{20} \leftarrow s_{10}s_8 + s_{13} + s_6 + 1$ ▷ 1M+1SR

9: $s_{21} \leftarrow s_{10}u_{c2}$, $s_{22} \leftarrow s_{14} + s_{21}$, $s_{23} \leftarrow s_{22}u_{c1}$ ▷ 2M

10: $s_{24} \leftarrow s_{12} + (s_{10} + s_{22})(u_{c1} + u_{c2}) + s_{21} + s_{23}$, $s_{25} \leftarrow s_{24}u_{c0}$ ▷ 2M

11: $v_{c2} \leftarrow s_{15} + s_{23} + (s_{10} + s_{24})(u_{c0} + u_{c2}) + s_{21} + s_{25}$ ▷ 1M

12: $s_{26} \leftarrow \mathrm{TR}(u_{c1}(u_{c2}(s_{17} + f_5) + v_{c2}^2 + u_{c0}))$ ▷ 2M+1S+1TR

13: **if** $s_{26} = 1$ **then**

14: $s_{20} \leftarrow s_{20} + 1$, $s_{27} \leftarrow s_{10}\sqrt{s_8}$, $s_{16} \leftarrow s_{16} + s_{27}^2$ ▷ 1M+1S+1SR

15: $u_{c2} \leftarrow u_{c2} + s_{10}$, $s_{28} \leftarrow s_{10}\sqrt{s_1}$, $u_{c1} \leftarrow u_{c1} + s_{28}$ ▷ 1M+1SR

16: $u_{c0} \leftarrow u_{c0} + s_{27}$, $s_{21} \leftarrow s_{21} + s_9$, $s_{23} \leftarrow s_{23} + s_{22}s_{28}$ ▷ 1M

17: $s_{24} \leftarrow s_{24} + s_{10}(s_{22} + s_{28})$, $s_{25} \leftarrow s_{24}u_{c0}$ ▷ 2M

18: $v_{c2} \leftarrow s_{15} + s_{28}^2 + s_{23} + (s_{10} + s_{24})(u_{c0} + u_{c2}) + s_{21} + s_{25}$ ▷ 1M+1S

19: **end if**

20: $v_{c1} \leftarrow s_{20} + (s_{24} + s_{22})(u_{c0} + u_{c1}) + s_{23} + s_{25}$, $v_{c0} \leftarrow s_{16} + s_{25}$ ▷ 1M

21: **return** $[x^3 + u_{c2}x^2 + u_{c1}x + u_{c0}, v_{c2}x^2 + v_{c1}x + v_{c0}]$

▷ 1I+25M+4S+7SR+1HT+1TR

We have a worst-case cost of 1I+25M+4S+7SR+1HT+1TR, which compares quite well with the doubling cost of 1I+44M+6S of [GKP04].

However, the conditional block of lines 13 to 19 is only used when the initial "choice" of the root of $z^2 + z + s_5 = 0$ (i.e. $\mathrm{HT}(s_5)$ rather than $\mathrm{HT}(s_5) + 1$) is incorrect and the variables computed afterwards must be corrected. This means that the 6M+2S+2SR associated to that correction in the conditional block will only be needed half of the time (on average), and the average cost of the halving operation becomes 1I+22M+3S+6SR+1HT+1TR.

4.6.3. Type IIa: $h(x) = x^2 + x + 1$

According to Section 4.4, curves of Type IIa are of the form

$$C : y^2 + (x^2 + x + 1)y = f_7x^7 + f_5x^5 + f_4x^4 + f_1x + f_0, \tag{4.15}$$

where $f_4 \in \mathbb{F}_2$ and $\mathrm{TR}(f_7) \cdot \mathrm{TR}(f_5) = 0$. The Picard group of these curves has precisely one divisor class of order 2, which is of the form $[h, v_h] = [x^2 + x + 1, v_h]$.

Theorem 4.5. Let $\overline{D}_a = [u_a, v_a]$ be a divisor class in $\mathrm{Pic}^0(C)$. If $\deg(u_a) = 3$, then $\overline{D}_a$ can be halved if and only if $\mathrm{TR}(f_7u_{a0} + f_4 + u_{a2}(f_7(u_{a1} + u_{a2}^2) + f_5 + f_7)) = 0$. If $\deg(u_a) = 2$, then $\overline{D}_a$ can be halved if and only if $\mathrm{TR}(u_{a1}(f_7(u_{a1}^2 + u_{a0}) + f_5 + f_7)) = 0$. If $\deg(u_a) = 1$, then $\overline{D}_a$ can be halved if and only if $\mathrm{TR}(f_4 + u_{a0}(f_7u_{a0}^2 + f_5 + f_7)) = 0$.

Proof. We use the same approach as in Theorem 4.2. Note that some of the formulas require solving two quadratic equations. In those cases, it is easy to verify that changing the root of the first quadratic equation changes the trace of the constant term of the second quadratic equation by 1, so only one of the two roots of the first quadratic equation allows us to compute an $\mathbb{F}_{2^d}$-rational preimage. □

Corollary 4.6. The Picard group of the curve C given by (4.15) has order $2r$, where r is odd, if and only if $\mathrm{TR}(f_7) \neq \mathrm{TR}(f_5)$.

Proof. The Picard group of the curve C has exactly one divisor class of order 2, namely $[x^2 + x + 1, v_h]$. The order of the Picard group is divisible by 4 if and only if $[x^2 + x + 1, v_h]$ can be halved. From Theorem 4.5, this is possible if and only if $\mathrm{TR}(f_5 + f_7) = 0$ and since $\mathrm{TR}(f_7) \cdot \mathrm{TR}(f_5) = 0$ we find that $\mathrm{Pic}^0(C)$ has a divisor class of order 4 if and only if $\mathrm{TR}(f_7) = \mathrm{TR}(f_5) = 0$. □

Observe that if $\overline{D}_{c_1}$ and $\overline{D}_{c_2}$ are the two preimages of $\overline{D}_a$ under the doubling, then $\overline{D}_{c_1} - \overline{D}_{c_2} = [x^2 + x + 1, v_h] = \overline{D}_{c_2} - \overline{D}_{c_1}$, i.e. the difference of two preimages is the unique divisor class of order 2. This observation allows us to distinguish the different special cases.

Remark 4.7. Let $\overline{D}_a = [u_a, v_a]$ be a divisor class in $\mathrm{Pic}^0(C)$ that can be halved and $\overline{D}_c = [u_c, v_c] = [\frac{1}{2}]\overline{D}_a$ its preimage (under the doubling) of odd order.

(1a) If $\deg(u_a) = 3$ and $v_{a2}^2+v_{a2}+u_{a2}(f_5+u_{a1}f_7+u_{a2}^2f_7)+\sqrt{u_{a2}f_7}+f_4+u_{a0}f_7 \neq 0$, then $\deg(u_c) = 3$ and we are in case HLV33.

(1b) If $\deg(u_a) = 3$ and $v_{a2}^2+v_{a2}+u_{a2}(f_5+u_{a1}f_7+u_{a2}^2f_7)+\sqrt{u_{a2}f_7}+f_4+u_{a0}f_7 = 0$, then $\deg(u_c) = 2$ or 3 and we are in case HLV32 or HLV33.

(2a) If $\deg(u_a) = 2$ and $u_{a1} \neq 0$, then $\deg(u_c) = 2$ and we are in case HLV23.

(2b) If $\deg(u_a) = 2$ and $u_{a1} = 0$, then $\deg(u_c) = 1$ or 2 (with $u_{c2} = u_{c1} = u_{c0} + 1$ in the second case) and we are in case HLV21 or HLV23.

(3) If $\deg(u_a) = 1$, then $\deg(u_c) = 3$ and we are in case HLV13.

Algorithm 34 (HLV33, $h(x) = x^2+x+1$, $f(x) = f_7x^7+f_5x^5+f_4x^4+f_1x+f_0$)

INPUT: $\overline{D} = [x^3 + u_{a2}x^2 + u_{a1}x + u_{a0}, v_{a2}x^2 + v_{a1}x + v_{a0}]$
OUTPUT: $[\frac{1}{2}]\overline{D} = [x^3 + u_{c2}x^2 + u_{c1}x + u_{c0}, v_{c2}x^2 + v_{c1}x + v_{c0}]$

$s_0 \leftarrow u_{a2}f_7$, $s_1 \leftarrow \sqrt{s_0}$, $s_2 \leftarrow (s_1 + f_5)f_7^{-1} + u_{a1}$ ▷ 2M+1SR
$s_3 \leftarrow s_1u_{a2} + v_{a2} + 1$, $s_4 \leftarrow s_1u_{a1} + v_{a1} + 1$, $s_5 \leftarrow s_1u_{a0} + v_{a0} + 1$ ▷ 3M
$s_6 \leftarrow s_2s_0$, $s_7 \leftarrow (s_6 + s_3^2 + s_3 + s_1 + f_4)f_7^{-1} + u_{a0}$, $s_8 \leftarrow s_7^{-1}$ ▷ 1I+2M+1S
$s_9 \leftarrow (s_7s_0 + s_4 + s_3 + s_1)f_7^{-1} + s_2u_{a1}$, $s_{10} \leftarrow f_7s_7$, $s_{11} \leftarrow \mathrm{HT}(s_{10})$ ▷ 4M+1HT
$s_{12} \leftarrow s_{11}s_8$, $s_{13} \leftarrow (f_7s_2 + s_1 + f_5 + s_{12})s_7$, $s_{14} \leftarrow \mathrm{TR}(s_{13})$ ▷ 3M+1TR
if $s_{14} = 1$ **then**
    $s_{11} \leftarrow s_{11} + 1$, $s_{12} \leftarrow s_{12} + s_8$, $s_{13} \leftarrow s_{13} + 1$
end if
$s_{15} \leftarrow \mathrm{HT}(s_{13})$, $s_{16} \leftarrow s_{15}s_8$, $s_{17} \leftarrow s_{12}s_2 + s_1$ ▷ 2M+1HT
$s_{18} \leftarrow s_{16}s_2 + s_3 + 1 + s_{11}$, $s_{19} \leftarrow s_{12}s_9 + s_4 + 1 + s_{15}$ ▷ 2M
$s_{20} \leftarrow (s_{12} + s_8 + f_7)f_7^{-2}$, $s_{21} \leftarrow (s_{17} + s_{16} + s_{12} + f_5)s_{20}$ ▷ 2M
$u_{c2} \leftarrow \sqrt{s_{21}}$, $s_{22} \leftarrow (s_{19} + s_{18} + s_{17})s_{20}$, $u_{c1} \leftarrow \sqrt{s_{22}}$ ▷ 1M+2SR
$s_{24} \leftarrow (s_{23} + s_{19} + f_1)s_{20}$, $u_{c0} \leftarrow \sqrt{s_{24}}$ ▷ 1M+1SR
$s_{25} \leftarrow \mathrm{TR}(u_{c0}f_7 + f_4 + u_{c2}(f_5 + f_7(u_{c1} + s_{21} + 1)))$ ▷ 3M+1TR
if $s_{25} = 1$ **then**
    $s_{16} \leftarrow s_{16} + s_8$, $s_{26} \leftarrow s_8s_2$, $s_{18} \leftarrow s_{18} + s_{26}$, $s_{19} \leftarrow s_{19} + 1$ ▷ 1M
    $s_{27} \leftarrow s_8s_9$, $s_{23} \leftarrow s_{23} + s_{27}$, $u_{c0} \leftarrow u_{c0} + \sqrt{(s_{27} + 1)s_{20}}$ ▷ 2M+1SR
    $u_{c2} \leftarrow u_{c2} + \sqrt{s_8s_{20}}$, $u_{c1} \leftarrow u_{c1} + \sqrt{(s_{26} + 1)s_{20}}$ ▷ 2M+2SR
end if
$s_{28} \leftarrow s_{12}u_{c2}$, $s_{29} \leftarrow s_{16} + s_{28}$, $s_{30} \leftarrow s_{29}u_{c1}$ ▷ 2M

21: $s_{31} \leftarrow s_{17} + (s_{12} + s_{29})(u_{c2} + u_{c1}) + s_{28} + s_{30}$, $s_{32} \leftarrow s_{31}u_{c0}$ ▷ 2M

22: $v_{c0} \leftarrow s_{23} + s_{32}$, $v_{c1} \leftarrow s_{19} + (s_{29} + s_{31})(u_{c1} + u_{c0}) + s_{30} + s_{32}$ ▷ 1M

23: $v_{c2} \leftarrow s_{18} + (s_{12} + s_{31})(u_{c2} + u_{c0}) + s_{28} + s_{32} + s_{30}$ ▷ 1M

24: **return** $[x^3 + u_{c2}x^2 + u_{c1}x + u_{c0},\ v_{c2}x^2 + v_{c1}x + v_{c0}]$

▷ 1I+36M+1S+7SR+2HT+2TR

We note that in these formulas a division by $s_{12}+f_7$ would normally be required to compute s_{21}, s_{22} and s_{24}. However, s_{12} is a root of $s_7z^2+z+f_7=0$ (since $s_{12}=s_{11}/s_7$), so $s_7(s_{12}+f_7)(s_{12}+f_7+1/s_7)=s_7s_{12}^2+s_7f_7^2+s_{12}+f_7=s_7f_7^2$. We can therefore replace divisions by $s_{12}+f_7$ with multiplications by $(s_{12}+f_7+1/s_7)f_7^2=s_{20}$, replacing the inverse by a single multiplication.

We therefore have a worst-case cost of 1I+36M+1S+7SR+2HT+2TR, which compares quite well with the doubling cost of 1I+52M+8S of [GKP04].

Conditional line 7 has very little impact on the overall cost, but the conditional block of lines 15 to 19 has a noticeable cost. However, it is only used when the initial "choice" of the root of $z^2+z+s_{13}=0$ (i.e. $\mathrm{HT}(s_{13})$ rather than $\mathrm{HT}(s_{13})+1$) is incorrect and the variables computed afterwards must be corrected. This means that the cost of 5M+3SR associated to that correction will only be needed half of the time (on average). The average cost of the halving operation becomes 1I+33.5M+1S+5.5SR+2HT+2TR.

4.6.4. Type Ia: $h(x)=x^3+x+h_0$ irreducible

According to Section 4.4, curves of Type Ia are of the form

$$C: y^2+(x^3+x+h_0)y=f_7x^7+x^6+f_2x^2+f_1x+f_0, \tag{4.16}$$

where x^3+x+h_0 is irreducible over $\mathbb{F}_{2^d}$ and $f_6\in\mathbb{F}_2$. The Picard group of these curves has precisely one divisor class of order 2, which is of the form $[h,v_h]=[x^3+x+h_0,v_h]$.

Proposition 4.8. If the polynomial x^3+x+h_0 is irreducible over $\mathbb{F}_{2^d}$, then the equation $x^4+x^2+h_0x+a=0$ has exactly one root in $\mathbb{F}_{2^d}$ for each $a\in\mathbb{F}_{2^d}$.

Proof. Since x^4 and x^2 act linearly in fields of characteristic 2, the operator $T(x)=x^4+x^2+h_0x$ is linear. We also note that the roots of $T(x)=0$ are 0, ζ_1, ζ_2 and ζ_3, where the ζ_i are the roots of $x^3+x+h_0=0$ in $\mathbb{F}_{2^{d^3}}\setminus\mathbb{F}_{2^d}$ (since x^3+x+h_0 is irreducible). Because of this, for any $a\in\mathbb{F}_{2^d}$ there cannot exist more than one $\mathbb{F}_{2^d}$-rational root, otherwise we would have two $\mathbb{F}_{2^d}$-rational roots of $T(x)=0$. To each element $\alpha\in\mathbb{F}_{2^d}$ we can associate a polynomial of the form $x^4+x^2+h_0x+a$, namely with $a=T(\alpha)$, all of which have exactly one $\mathbb{F}_{2^d}$-rational root. □

Note that $T(x)=x^4+x^2+h_0x$ being a linear operator also allows us to compute the $\mathbb{F}_{2^d}$-rational root. We first compute the images of $T(e_i)$ for every e_i in the basis used to represent field elements, which gives us a system of linear equations (that can be used to describe the image of every field element). By inverting this system, we can precompute the roots x_i of $x^4+x^2+h_0x+e_i=0$. For any given $a\in\mathbb{F}_{2^d}$, $a=\sum_{i=0}^{n-1}a_ie_i$ (with $a_i\in\mathbb{F}_2$), we can then compute the root x_a of $x^4+x^2+h_0x+a=0$ as $x_a=\sum_{i=0}^{n-1}a_ix_i$. With a little more work (computing the roots for all blocks of w bits), it becomes possible to compute roots of the quartic

in time QR at least as fast as a multiplication. In fact, this method is equivalent to what is used to compute half-traces, so QR $\approx$ HT $\leq$ M.

Remark 4.9. In the case where $h_0 = 1$, we can express the root of $x^4+x^2+x+a = 0$ as the "two-third-trace" of a: if $n \equiv 1 \bmod 3$, we let $x_a = \mathrm{TR}(a) - \sum_{i=0}^{\frac{n-4}{3}} a^{2^{3i+1}}$, and if $n \equiv 2 \bmod 3$, we let $x_a = \mathrm{TR}(a) - \sum_{i=0}^{\frac{n-2}{3}} a^{2^{3i}}$.

Theorem 4.10. Let $\overline{D}_a = [u_a, v_a]$ be a divisor class in $\mathrm{Pic}^0(C)$. If $\deg(u_a) = 3$, then $\overline{D}_a$ can be halved if and only if $\mathrm{TR}(f_7 u_{a2} + f_6) = 0$. If $\deg(u_a) = 2$, then $\overline{D}_a$ can be halved if and only if $\mathrm{TR}(f_7 u_{a1}) = 0$. If $\deg(u_a) = 1$, then $\overline{D}_a$ can be halved if and only if $\mathrm{TR}(f_7 u_{a0} + f_6) = 0$.

Proof. As in Theorem 4.2. □

Corollary 4.11. The Picard group of the curve C given by (4.16) has order $2r$, where r is odd, if and only if $f_6 = 1$.

Proof. The Picard group of the curve C has exactly one divisor class of order 2, namely $[x^3 + x + h_0, v_h]$. The order of the Picard group is divisible by 4 if and only if $[x^3 + x + h_0, v_h]$ can be halved. From Theorem 4.10, this is possible if and only if $\mathrm{TR}(f_6) = 0$. Since $f_6 \in \mathbb{F}_2$, we find that $\mathrm{Pic}^0(C)$ has a divisor class of order 4 if and only if $f_6 = 0$. □

Observe that if $\overline{D}_{c_1}$ and $\overline{D}_{c_2}$ are the two preimages of $\overline{D}_a$ under the doubling, then $\overline{D}_{c_1} - \overline{D}_{c_2} = [x^3 + x + h_0, v_h] = \overline{D}_{c_2} - \overline{D}_{c_1}$, i.e. the difference of two preimages is the unique divisor class of order 2. This observation allows us to distinguish the different special cases.

Remark 4.12. Let $\overline{D}_a = [u_a, v_a]$ be a divisor class in $\mathrm{Pic}^0(C)$ that can be halved and $\overline{D}_c = [u_c, v_c] = [\frac{1}{2}]\overline{D}_a$ its preimage (under the doubling) of odd order.

(1a) If $\deg(u_a) = 3$ and

$$\begin{aligned}(1 + u_{a1})(v_{a1} + u_{a0} f_7 + v_{a2}^2 + u_{a2} v_{a2} + (u_{a2}^2 + u_{a1})(1 + u_{a2} f_7)) \\ +(v_{a1} + u_{a0} f_7 + v_{a2}^2 + u_{a2} v_{a2} + (u_{a2}^2 + u_{a1})(1 + u_{a2} f_7))^2 \\ +u_{a1} + (1 + u_{a2} f_7)(1 + u_{a1}^2) \neq 0,\end{aligned}$$

then $\deg(u_c) = 3$ and we are in case HLV33.

(1b) If $\deg(u_a) = 3$ and

$$\begin{aligned}(1 + u_{a1})(v_{a1} + u_{a0} f_7 + v_{a2}^2 + u_{a2} v_{a2} + (u_{a2}^2 + u_{a1})(1 + u_{a2} f_7)) \\ +(v_{a1} + u_{a0} f_7 + v_{a2}^2 + u_{a2} v_{a2} + (u_{a2}^2 + u_{a1})(1 + u_{a2} f_7))^2 \\ +u_{a1} + (1 + u_{a2} f_7)(1 + u_{a1}^2) = 0,\end{aligned}$$

then $\deg(u_c) = 2$ or 3 and we are in case HLV32 or HLV33.

(2a) If $\deg(u_a) = 2$ and $u_{a1} \neq 0$, then $\deg(u_c) = 2$ and we are in case HLV23.

(2b) If $\deg(u_a) = 2$ and $u_{a1} = 0$, then $\deg(u_c) = 1$ or 2 and we are in case HLV21 or HLV23.

(3) If $\deg(u_a) = 1$, then $\deg(u_c) = 3$ and we are in case HLV13.

We obtain the following halving formulas:

Algorithm 35 (HLV33, $h(x) = x^3 + x + h_0$ irreducible, $f(x) = f_7x^7 + x^6 + f_2x^2 + f_1x + f_0$)

INPUT: $\overline{D} = [x^3 + u_{a2}x^2 + u_{a1}x + u_{a0}, v_{a2}x^2 + v_{a1}x + v_{a0}]$

OUTPUT: $[\frac{1}{2}]\overline{D} = [x^3 + u_{c2}x^2 + u_{c1}x + u_{c0}, v_{c2}x^2 + v_{c1}x + v_{c0}]$

1: $s_0 \leftarrow \mathrm{HT}(f_7u_{a2} + 1)$, $s_1 \leftarrow v_{a2} + u_{a2}(s_0 + 1)$ ▷ 2M+1HT

2: $s_2 \leftarrow v_{a1} + 1 + u_{a1}(s_0 + 1)$, $s_3 \leftarrow v_{a0} + h_0 + u_{a0}(s_0 + 1)$ ▷ 2M

3: $s_4 \leftarrow s_1f_7^{-1} + u_{a1}$, $s_5 \leftarrow (s_2 + s_0 + s_1^2)f_7^{-1} + u_{a0} + s_4u_{a2}$ ▷ 3M+1S

4: $s_6 \leftarrow (s_1 + (s_4 + 1)f_7)s_5 + (s_5f_7)^2$, $s_7 \leftarrow \mathrm{QR}(s_6)$ ▷ 3M+1S+1QR

5: $s_8 \leftarrow s_7^2$, $s_9 \leftarrow s_8s_5$, $s_{10} \leftarrow s_9^{-1}$, $s_{11} \leftarrow s_{10}s_8$, $s_{12} \leftarrow s_{10}s_5^2$ ▷ 1I+3M+2S

6: $s_{13} \leftarrow s_7s_{11}$, $s_{14} \leftarrow s_{13}^2s_5 + f_7$, $s_{15} \leftarrow (s_4 + 1)s_{14} + (h_0 + s_5)s_{13} + s_1$ ▷ 4M+1S

7: $s_{16} \leftarrow s_{12}s_{15}$, $s_{17} \leftarrow \mathrm{TR}(s_{16}f_7^2)$ ▷ 2M+1TR

8: **if** $s_{17} = 0$ **then**

9: $s_0 \leftarrow s_0 + 1$, $s_1 \leftarrow s_1 + u_{a2}$, $s_2 \leftarrow s_2 + u_{a1}$, $s_3 \leftarrow s_3 + u_{a0}$

10: $s_4 \leftarrow s_4 + u_{a2}f_7^{-1}$, $s_5 \leftarrow s_5 + (u_{a1} + 1)f_7^{-1}$ ▷ 2M

11: $s_6 \leftarrow (s_1 + (s_4 + 1)f_7)s_5 + (s_5f_7)^2$, $s_7 \leftarrow \mathrm{QR}(s_6)$ ▷ 3M+1S+1QR

12: $s_8 \leftarrow s_7^2$, $s_9 \leftarrow s_8s_5$, $s_{10} \leftarrow s_9^{-1}$, $s_{11} \leftarrow s_{10}s_8$ ▷ 1I+2M+1S

13: $s_{12} \leftarrow s_{10}s_5^2$, $s_{13} \leftarrow s_7s_{11}$, $s_{14} \leftarrow s_{13}^2s_5 + f_7$ ▷ 3M+2S

14: $s_{15} \leftarrow (s_4 + 1)s_{14} + (h_0 + s_5)s_{13} + s_1$, $s_{16} \leftarrow s_{12}s_{15}$ ▷ 3M

15: **end if**

16: $s_{18} \leftarrow (s_3 + s_1 + s_0h_0)f_7^{-1} + s_4u_{a1} + s_5u_{a2}$, $s_{19} \leftarrow s_{13}s_4$ ▷ 5M

17: $s_{20} \leftarrow s_0 + 1 + s_{19}$, $s_{21} \leftarrow (s_{14} + s_{13})(s_5 + s_4)$, $s_{22} \leftarrow s_{14}s_5$ ▷ 2M

18: $s_{23} \leftarrow s_1 + s_{19} + s_{21} + s_{22}$, $s_{24} \leftarrow s_2 + 1 + s_{13}s_{18} + s_{22}$ ▷ 1M

19: $s_{25} \leftarrow h_0 + s_3 + s_{14}s_{18}$, $s_{26} \leftarrow (s_{23} + s_{20}h_0 + s_{25})s_{12}$ ▷ 3M

20: $s_{27} \leftarrow (f_1 + s_{24}h_0 + s_{25})s_{12}$, $u_{c2} \leftarrow \sqrt{s_{16}}$, $u_{c1} \leftarrow \sqrt{s_{26}}$ ▷ 2M+2SR

21: $u_{c0} \leftarrow \sqrt{s_{27}}$, $s_{28} \leftarrow s_{13}u_{c2}$, $s_{29} \leftarrow s_{14} + s_{28}$, $s_{30} \leftarrow s_{29}u_{c1}$ ▷ 2M+1SR

22: $s_{31} \leftarrow s_{20} + (s_{13} + s_{29})(u_{c2} + u_{c1}) + s_{28} + s_{30}$, $s_{32} \leftarrow s_{31}u_{c0}$ ▷ 2M

23: $v_{c0} \leftarrow s_{25} + s_{32}$, $v_{c1} \leftarrow s_{24} + (s_{29} + s_{31})(u_{c1} + u_{c0}) + s_{30} + s_{32}$ ▷ 1M

24: $v_{c2} \leftarrow s_{23} + (s_{13} + s_{31})(u_{c2} + u_{c0}) + s_{28} + s_{32} + s_{30}$ ▷ 1M
25: **return** $[x^3 + u_{c2}x^2 + u_{c1}x + u_{c0},\ v_{c2}x^2 + v_{c1}x + v_{c0}]$
▷ 2I+51M+9S+3SR+1TR+1HT+2QR

We note that in these formulas a division by $s_{14}+f_7$ would normally be required to compute s_{16}, s_{26} and s_{27}. However, $s_{14} + f_7 = s_{13}^2 s_5$, which we can compute as $s_5/(s_{13}s_5)^2$. Since $s_{13} = s_7/s_5$, $1/(s_{13}s_5) = 1/s_7$ and we can combine this inverse with the computation of $1/s_5$. As a result, we can compute both inverses using only 1I+3M+1S.

We therefore have a worst-case cost of 2I+49M+9S+3SR+1HT+1TR+2QR, which compares well with the doubling cost of 1I+63M+9S of Guyot, Kaveh and Patankar [GKP04], as long as inversion costs are not too high.

However, the conditional block of lines 10 to 17 is only used when the initial "choice" for the root of $z^2 + z + u_{a2} + 1 = 0$ (i.e. $\mathrm{HT}(u_{a2} + 1)$ rather than $\mathrm{HT}(u_{a2} + 1) + 1$) is incorrect and the variables computed afterwards must be corrected. This means that the 1I+13M+4S+1QR associated to that correction will only be needed half of the time (on average). The average cost of the halving operation becomes 1.5I+42.5M+7S+3SR+1HT+1TR+1.5QR.

Remark 4.13. There is another approach to "optimise" the formulas, limiting ourselves to no more than one inversion per halving. The idea consists of doing the computations for both roots of $z^2 + z + u_{a2} + 1 = 0$ together until the computations of the inverses, at which points the two inverses can be combined into one using Montgomery's trick (doing both in 1I+3M), after which we can use the normal branching approach. In this way, we get a worst-case cost of 1I+52M+9S+3SR+1HT+1TR+2QR, from which we expect to save 7M+2S when the first choice of the root is correct (half of the time). The final cost will increase whenever an inversion costs less than 12M+2S+1QR, making this approach unlikely to be useful with many implementations of the field arithmetic (for field sizes used on genus-3 curves at standard cryptographic security levels).

4.6.5. Discussion of the halving approach

To obtain the formulas in this section, we inverted Cantor's doubling algorithm rather than inverting the corresponding explicit formulas. Even though we used the general algorithm rather than the highly optimised version to obtain our formulas, we obtained operations that are more efficient.

At first glance, this could seem contradictory. After all, one of the main methods used in explicit formulas to produce such savings in comparison with Cantor's algorithm is through the merging of the composition and the first reduction step. This merging is completely ignored in our approach, but the resulting formulas are still faster.

At the same time, the halving formulas must include the cost coming from choosing the "wrong" roots of quadratic equations, which naturally increases as the quadratic equation is encountered earlier in the formula (since the condition to determine the "correct root" comes from whether or not the computed preimage can be halved again). As a consequence, one would expect the correction cost to increase as the degree of h also does, and in fact this is more or less what we observe ($h = x^2 + x + 1$ seems to be an exception to this rule of thumb). This means that the halving formulas get a higher penalty for selecting the correct preimage when the degree of h increases, and we do see this very clearly for the curves with h irreducible of degree 3.

Nevertheless, the absolute saving when comparing with the doubling seems to remain almost constant between the different curve types. In fact these apparent discrepancies come from the inherent difference between doubling and halving.

In the doubling, even with optimised explicit formulas, the composition step requires the computation of h^{-1} modulo u_a, or at least its almost-inverse, after which the reduction steps are relatively simple and straightforward. In fact, simply looking at the distribution of the cost in the different steps of the algorithm makes it quite clear that the composition, and in particular the computation of the almost-inverse, is one of the dominant factors.

For the halving, we work our way backwards through the reductions steps until we obtain $[u_0, v_0]$. In general, the cost of an "un-reduction" step may be higher than for the corresponding reduction step, but this increase is usually small. Once $[u_0, v_0]$ is known, computing u_c (the first polynomial of the output) only requires computing the square-root of u_0, while v_c is obtained by reducing v_0 modulo u_c.

These last two operations are quite inexpensive, requiring a total of 6M and 3SR, no matter what form the curve has. In comparison, the composition step, even when merged with the first reduction, requires the computation of h^{-1} modulo u_a (or an almost-inverse), which becomes much more costly as the degree of h increases. The savings obtained by switching from almost-inverse to modular reduction (from doubling to halving) are therefore much greater when h becomes more complicated, and easily compensate for any of the "inconveniences" of halving that we just described.

This also explains why the costs of doubling and halving are essentially identical when $h = 1$: in that case, h^{-1} comes for free and the optimised doubling does not merge the composition and first reduction, making doubling and halving perfect mirror images of each other.

4.7. Choice of secure curves

In the previous sections we have studied hyperelliptic curves of genus-3 over binary fields, and provided efficient arithmetic on these curves. Now we will discuss the security of these curves and suggest curve parameters and field sizes to make the

DLP on these curves intractable against currently known attacks.

For genus-3 index calculus attacks are not applicable. To avoid the other types of attacks we restrict the base field to $\mathbb{F}_{2^d}$, where d is prime (this is to avoid Weil descent attacks, see Section A.6 for details), request that the embedding degree k is at least 3000 (to avoid Frey-Rück attacks, see Section A.5) and that the order of $\mathrm{Pic}^0_{\mathbb{F}_{2^d}}(C)$ is almost prime and sufficiently large. Due to the Hasse-Weil bound (Theorem 2.15) the size of the Picard group of a hyperelliptic curve of genus 3 over $\mathbb{F}_{2^d}$ is at most $(\sqrt{2^d}+1)^6$ which is approximately 2^{3d}. Since the best known attacks for genus 3 are square-root attacks, e.g. baby-step giant-step (Section A.2) and Pollard rho (Section A.3), the security level of these curves is $(3/2)d$ bit.

According to the latest recommendations 80-bit security (i.e. $d \approx 66$) is borderline and 112 bit (i.e. $d \approx 75$) offer medium-term protection whereas 128 bit (i.e. $d \approx 83$) are good for long-term secrecy (see [Gir08] and the ECRYPT recommendations [Nae08] for more details). Suitable primes in these ranges are $d = 67$, $d = 79$ and $d = 83$.

5. Edwards Curves

Edwards in [Edw07] introduced a new normal form for elliptic curves. He showed that every elliptic curve over a field k of characteristic different from 2 can be written in this normal form over an extension of k. Bernstein and Lange in [BL07b] studied the use of *Edwards curves* for cryptography and provided efficient point addition and point doubling formulas. It turns out that the arithmetic on Edwards curves is even faster than on Jacobian-form elliptic curves introduced in [CC87] which were considered the fastest so far. Soon after their first paper on Edwards curves, Bernstein and Lange improved their own results by introducing *inverted Edwards coordinates* [BL07c] which provide even faster addition formulas on Edwards curves.

In this chapter, we give an introduction to Edwards curves, their addition law and we show how inverted Edwards coordinates can make this addition faster. After that, we demonstrate how the concept of Edwards curves can be generalised to cover a larger family of curves. In particular, we show that every elliptic curve in Montgomery form is birationally equivalent to a *twisted Edwards curve* (see Section 5.3.1). We also give the group law for this type of curve.

Furthermore, we develop tripling formulas for points on Edwards curves. With that, we show that for single-scalar multiplication using double-base chains (with basis $\{2, 3\}$) there is no gain in speed compared to traditional methods based on double-and-add.

In this thesis, we consider Edwards and twisted Edwards curves over non-binary fields only. For the characteristic-2 case, we refer the reader to [BLRF08], in which binary Edwards curves are introduced and discussed.

Section 5.2 is a review of Edwards curves, based on the papers by Edwards [Edw07] and Bernstein and Lange [BL07b, BL07c]. The results on the tripling formulas in Section 5.4 and on optimising scalar multiplication in Section 5.5 are new and are joint work with Bernstein, Lange and Peters [BBLP07]. The results on twisted Edwards curves in Section 5.3 are new and are joint work with Bernstein, Joye, Lange and Peters [BBJ+08].

Notation

Note that in this chapter we use the same notation for the operation counts as in the two previous chapters. A field multiplication is denoted by M, an inversion by I, a squaring by S, the extraction of a square root by SR, a field addition by add and a multiplication by 2 by 1times2. The letter D stands for a multiplication by

a curve parameter. We treat those multiplications separately because they are cheaper than normal multiplications if the curve parameter has small height.

Before we begin to explain Edwards curves, we need to present some facts about birational equivalence and desingularisation of curves.

5.1. Birational equivalence and desingularisation

Definition 5.1 (Rational map)**.** Let k be any field. Let C_1 and C_2 be two curves in $\mathbb{P}^2(k)$. A *rational map* from C_1 to C_2 is of the form

$$\varphi : C_1 \to C_2$$
$$\varphi = (f_0, f_1, f_2),$$

where $f_0, f_1, f_2 \in \overline{k}(C_1)$ have the property that for every point $P \in C_1$ at which f_0, f_1, f_2 are all defined,

$$\varphi(P) = (f_0(P) : f_1(P) : f_2(P)) \in C_2. \tag{5.1}$$

Remark 5.2 ([Sil86], page 15)**.** Note that a rational map $\varphi : C_1 \to C_2$ is not necessarily a function on all of C_1. However, it is sometimes possible to evaluate $\varphi(P)$ at points P of C_1 where some f_i is not regular by replacing each f_i with gf_i for an appropriate $g \in \overline{k}(C_1)$.

Definition 5.3 (Birational equivalence, isomorphic)**.** Two curves C_1 and C_2 are *birationally equivalent* if there exist two rational maps $\varphi : C_1 \to C_2$ and $\psi : C_2 \to C_1$ with $\varphi \circ \psi = \mathrm{id}_{C_1}$ and $\psi \circ \varphi = \mathrm{id}_{C_2}$.

The curves are *isomorphic* if in addition φ is regular on all points of C_1 and ψ is regular on all points on C_2.

For the following statements, it is necessary that the curves are absolutely irreducible.

Theorem 5.4. Every projective curve C is birationally equivalent to a (not necessarily plane) projective smooth curve, which is called *desingularisation* of C.

The next theorem justifies that we speak of isomorphism classes of desingularisations.

Theorem 5.5. Let C_1 and C_2 be two projective and smooth curves. If they are birationally equivalent, then they are isomorphic.

From now on, due to the existence of the desingularisation we can extend the definition of the genus to singular curves.

Definition 5.6. Let C be a curve. We call the genus of C the genus of the desingularisation.

Proposition 5.7. If two curves are birationally equivalent, then they have the same genus.

5.2. Edwards curves

In this section, we give the definition of Edwards curves and investigate their properties. We provide the (affine) addition law and present projective and inverted Edwards coordinates together with explicit formulas for point addition and point doubling.

Definition 5.8. Let k be a field with $\mathrm{char}(k) \neq 2$. An *Edwards curve* over k is given by an equation of the form

$$E : x^2 + y^2 = 1 + dx^2y^2, \tag{5.2}$$

where $d \in k$ is not equal to 0 or 1. An affine point on an Edwards curve is given by a pair (x_1, y_1) such that x_1 and y_1 satisfy the curve equation.

If d equals 0, then (5.2) describes a circle of radius 1 which has genus 0. For $d = 1$, the equation factors as $(x^2 - 1)(y^2 - 1) = 0$. Hence, in both cases E is not an elliptic curve. To see that E is a non-singular curve in all other cases, we consider the two partial derivatives $2x = 2dxy^2$ and $2y = 2dyx^2$ of (5.2). For x and y not equal to 0 we get $1 = dy^2$ and $1 = dx^2$ which implies $d = 1/y^2$. Plugging this into the curve equation gives $x^2 + y^2 = 1 + x^2$ and thus $y = \pm 1$. It follows that $d = 1$ which is a contradiction. The only points on the curve with x-coordinate 0 are $(0, 1)$ and $(0, -1)$. In those cases the partial derivative with respect to y becomes $2 = 0$ or $-2 = 0$ which is false since $\mathrm{char}(k) \neq 2$. The points with y-coordinate 0 are $(-1, 0)$ and $(1, 0)$. With the same argument the partial derivative with respect to x does not hold true. We can conclude that it does indeed make sense to exclude exactly $0, 1$ for d in Definition 5.8.

Edwards in [Edw07] shows that (5.2) describes an elliptic curve, so the curve E has genus 1 and it is covered by Definition 2.1.

The following theorem states conditions under which (twists of) elliptic curves are birationally equivalent to curves in Edwards form.

Theorem 5.9. Let k be a field in which $2 \neq 0$. Let E be an elliptic curve over k such that the group $E(k)$ has an element of order 4. Then:

(1) There exists $d \in k \setminus \{0, 1\}$ such that the curve $x^2 + y^2 = 1 + dx^2y^2$ is birationally equivalent over k to a quadratic twist of E.

(2) If $E(k)$ has a unique element of order 2, then there is a non-square $d \in k$ such that the curve $x^2 + y^2 = 1 + dx^2y^2$ is birationally equivalent over k to a quadratic twist of E.

(3) If k is finite and $E(k)$ has a unique element of order 2, then there is a non-square $d \in k$ such that the curve $x^2 + y^2 = 1 + dx^2y^2$ is birationally equivalent over k to E.

Proof. See Theorem 2.1 in [BL07b]. □

5.2.1. Addition law and properties

For affine points, the addition law on an Edwards curve is given by

$$(x_1, y_1), (x_2, y_2) \mapsto \left(\frac{x_1 y_2 + y_1 x_2}{1 + d x_1 x_2 y_1 y_2}, \frac{y_1 y_2 - x_1 x_2}{1 - d x_1 x_2 y_1 y_2} \right). \tag{5.3}$$

If the curve parameter d is not a perfect square, then the addition law is complete, i.e. the addition formulas hold for any valid input without exception. The point $(0, 1)$ is the neutral element in the group of points on E. The negative of a point $P = (x_1, y_1)$ can be computed by reflecting it across the y-axis, so we have $-P = (-x_1, y_1)$. The point $(0, -1)$ has order 2. The points $(1, 0)$ and $(-1, 0)$ have order 4.

In the projective model, the Edwards curve equation is

$$(X^2 + Y^2)Z^2 = Z^4 + dX^2Y^2. \tag{5.4}$$

To find its points at infinity we put $Z = 0$ and look for points (X, Y) such that $0 = dX^2Y^2$. First we assume $X \neq 0$. To satisfy the equation, we need $Y = 0$. Thus the point $(1 : 0 : 0)$ is at infinity. Now we assume $Y \neq 0$. Analogously we require $X = 0$ and see that $(0 : 1 : 0)$ is the second point at infinity. There are no more points at infinity on the projective curve. To check whether the two points are singular, we dehomogenise the curve equation with respect to X and Y and check using the partial derivatives if the point $(0, 0)$ is singular on both affine curves. It turns out that both points are singularities on the projective curve, and after resolving them using blow-up techniques we get two further points of order 2 and two points of order 4 of which the minimal field of definition is $k(\sqrt{d})$. Therefore, there are three k-rational points of order 2 if and only if d is a square in k.

5.2.2. Projective Edwards coordinates

An affine point on an Edwards curve is given by a pair (x_1, y_1). To avoid inversions in the addition and doubling formulas, one usually homogenises the curve equation. This leads to

$$(X^2 + Y^2)Z^2 = (Z^4 + dX^2Y^2). \tag{5.5}$$

A point $(X_1 : Y_1 : Z_1)$ with $Z_1 \neq 0$ that satisfies this equation, corresponds to the affine point $(X_1/Z_1, Y_1/Z_1)$. The neutral element in projective coordinates is the point $(0 : 1 : 1)$.

Inversion-free formulas for addition and doubling in homogenised form were introduced by Bernstein and Lange [BL07b]. A general addition in Edwards coordinates takes 10M+1S+1D+7add, i.e. 10 field multiplications, 1 field squaring,

1 multiplication by the curve parameter d and 7 field additions. A doubling takes 3M+4S+6add.

For a collection of explicit formulas and operation counts for elliptic curves in various representations we refer to the Explicit-Formulas Database [BL07a].

5.2.3. Inverted Edwards coordinates

Another way of representing points on an Edwards curve is using *inverted Edwards coordinates* [BL07c]. This coordinate system allows inversion-free addition and doubling. The advantage of inverted Edwards coordinates is that the addition is faster than in the projective case. On the other hand the doubling is slightly more costly.

A projective point $(X_1 : Y_1 : Z_1)$ with $X_1, Y_1 \neq 0$ in inverted Edwards coordinates corresponds to the affine point $(Z_1/X_1, Z_1/Y_1)$. Normally, in projective coordinates one divides by the third coordinate Z_1. Using inverted Edwards coordinates it is the other way round; we divide Z_1 by X_1 and by Y_1. This is the reason for the name "inverted". The addition of two points in inverted Edwards coordinates costs 9M+1S+1D+7add, which is a speedup of 1M compared to standard projective Edwards coordinates. However, a doubling costs 3M+4S+1D+5add+1times2, and is thus a little more expensive than in projective Edwards coordinates. The addition and doubling formulas are provided below.

Algorithm 36 (Addition in inverted Edwards coordinates)

INPUT: Two points (X_1, Y_1, Z_1), (X_2, Y_2, Z_2)
OUTPUT: The point $(X_3, Y_3, Z_3) = (X_1, Y_1, Z_1) + (X_2, Y_2, Z_2)$

1: $A \leftarrow Z_1 Z_2$, $B \leftarrow dA^2$, $C \leftarrow X_1 X_2$, $D \leftarrow Y_1 Y_2$, $E \leftarrow CD$ ▷ 4M+1S+1D
2: $H \leftarrow C - D$, $I \leftarrow (X_1 + Y_1)(X_2 + Y_2) - C - D$ ▷ 1M+5add
3: $X_3 \leftarrow (E + B)H$, $Y_3 \leftarrow (E - B)I$, $Z_3 \leftarrow AHI$ ▷ 4M+2add
4: **return** (X_3, Y_3, Z_3) ▷ 9M+1S+1D+7add

Algorithm 37 (Doubling in inverted Edwards coordinates)

INPUT: The point (X_1, Y_1, Z_1)
OUTPUT: The point $(X_3, Y_3, Z_3) = [2](X_1, Y_1, Z_1)$

1: $A \leftarrow X_1^2$, $B \leftarrow Y_1^2$, $C \leftarrow A + B$, $D \leftarrow A - B$ ▷ 2S+2add
2: $E \leftarrow (X_1 + Y_1)^2 - C$, $Z_3 \leftarrow DE$, $X_3 \leftarrow CD$ ▷ 2M+1S+2add

3: $Y_3 \leftarrow E(C - 2dZ_1^2)$ ▷ 1M+1S+1D+1add+1times2
4: **return** (X_3, Y_3, Z_3) ▷ 3M+4S+1D+5add+1times2

5.3. Twisted Edwards curves

An Edwards curve has always a point of order 4. This restricts the number of elliptic curves in Edwards form over k. To cover more elliptic curves, we embed the set of Edwards curves in a larger set of elliptic curves with a similar shaped curve equation by introducing twisted Edwards curves. The work in this section is based on [BBJ+08].

In 1987, Montgomery [Mon87] introduced elliptic curves with curve equations of the form $Bv^2 = u^3 + Au^2 + u$ to speed up the Pollard and Elliptic Curve Methods of integer factorisation. These Montgomery curves turned out to have interesting properties. In this chapter, we will show that every twisted Edwards curve is birationally equivalent to an elliptic curve in Montgomery form, and vice versa. With this equivalence we bring the speed of the Edwards addition law to every elliptic curve in Montgomery form.

Definition 5.10. Let k be a field with $\mathrm{char}(k) \neq 2$. For two distinct non-zero elements $a, d \in k$ the *twisted Edwards curve* with coefficients a, d is given by the equation

$$\mathrm{E}_{\mathrm{E},a,d} : ax^2 + y^2 = 1 + dx^2y^2. \tag{5.6}$$

An Edwards curve is a twisted Edwards curve with $a = 1$. We will see later on in this section that a twisted Edwards curve is isomorphic to an Edwards curve if and only if a is a square in k.

For the purpose of Section 6.2 the twisted form is interesting since we will find and use curves over $\mathbb{Q}$ with coefficients d of small height, i.e. small numerator and denominator. The smallest integer congruent to d modulo n will usually have as many bits as n, so multiplications by d, as they appear in the addition and doubling formulas of inverted Edwards coordinates, are costly while multiplications by the numerator and denominator separately are cheap.

5.3.1. Montgomery curves and twisted Edwards curves

In this section, we give the definition and basic facts about Montgomery curves. Furthermore, we prove that every elliptic curve in Montgomery form over a non-binary field is birationally equivalent to a twisted Edwards curve.

Definition 5.11 (Montgomery curve)**.** Let k be a field with $\mathrm{char}(k) \neq 2$. Fix $A \in k \setminus \{-2, 2\}$ and $B \in k \setminus \{0\}$. The *Montgomery curve* with coefficients A and B is the curve

$$\mathrm{E}_{\mathrm{M},A,B} : Bv^2 = u^3 + Au^2 + u. \tag{5.7}$$

This form of the curve equation allows a very efficient arithmetic which relies on the computation of x-coordinates only. This technique was first introduced by Montgomery [Mon87] for fields with large characteristic.

In projective coordinates, point addition and point doubling on a Montgomery curve can be computed without the knowledge of any y-coordinate. Hence the advantages are very efficient addition and doubling formulas. A point addition requires 4M+2S, while a doubling requires 3M+2S (see Section 13.2.3 in [ACD+05]). The scalar multiplication $[n]P$ can be efficiently computed using the Montgomery ladder (cf. Section 13.2.3.d in [ACD+05]), but one still needs to recover the y-coordinate afterwards.

The drawback of Montgomery arithmetic is that some applications (e.g. the ElGamal signature scheme) require the computation of many y-coordinates, but recovering these coordinates is comparatively inefficient.

The next theorem states the birational equivalence between Montgomery curves and twisted Edwards curves over non-binary fields.

Theorem 5.12. Let k be a field with $\mathrm{char}(k) \neq 2$ and let a, d be distinct non-zero elements of k. Let $u = (1+y)/(1-y)$ and $v = u/x$. Then the map $(x, y) \mapsto (u, v)$ with inverse $(u, v) \mapsto (u/v, (u-1)/(u+1))$ is a birational equivalence over k from the twisted Edwards curve $ax^2 + y^2 = 1 + dx^2y^2$ to the Montgomery curve $Bv^2 = u^3 + Au^2 + u$, where $B = 4/(a-d)$ and $A = 2(a+d)/(a-d)$.

Conversely, every Montgomery curve over k is birationally equivalent over k to a twisted Edwards curve. Specifically, fix $A \in k \setminus \{-2, 2\}$ and $B \in k \setminus \{0\}$. Then the Montgomery curve $\mathrm{E}_{\mathrm{M},A,B}$ is birationally equivalent to the twisted Edwards curve $\mathrm{E}_{\mathrm{E},a,d}$, where $a = (A+2)/B$ and $d = (A-2)/B$.

Proof. We show that $Bv^2 - u^3 - Au^2 - u = 0$ holds if u, v, A and B are chosen as above and $(x, y) \in k \times k$ is a point on the twisted Edwards curve.

$$\begin{aligned} Bv^2 - u^3 - Au^2 - u &= \frac{4(1+y)^2}{(a-d)x^2(1-y)^2} - \frac{(1+y)^3}{(1-y)^3} - \frac{2(a+d)(1+y)^2}{(a-d)(1-y)^2} \\ &\quad - \frac{1+y}{1-y} \\ &= \frac{-4(1+y)(ax^2 + y^2 - 1 - dx^2y^2)}{(a-d)x^2(1-y)^3} = 0 \end{aligned}$$

since $ax^2 + y^2 = 1 + dx^2y^2$.

The exceptional cases $y = 1$ and $x = 0$ occur for only finitely many points (x, y) on $\mathrm{E}_{\mathrm{E},a,d}$. Conversely, we have $x = u/v$ and $y = (u-1)/(u+1)$ and the exceptional cases $v = 0$ and $u = -1$ occur for only finitely many points (u, v) on $\mathrm{E}_{\mathrm{M},A,B}$.

To proof the second part of the theorem, note that a and d are defined, since $B \neq 0$. Note further that $a \neq 0$ since $A \neq -2$; $d \neq 0$ since $A \neq 2$; and $a \neq d$.

Thus $\mathrm{E}_{\mathrm{E},a,d}$ is a twisted Edwards curve. Furthermore

$$2\frac{a+d}{a-d} = 2\frac{\frac{A+2}{B}+\frac{A-2}{B}}{\frac{A+2}{B}-\frac{A-2}{B}} = A \quad \text{and} \quad \frac{4}{(a-d)} = \frac{4}{\frac{A+2}{B}-\frac{A-2}{B}} = B.$$

Hence $\mathrm{E}_{\mathrm{E},a,d}$ is birationally equivalent to $\mathrm{E}_{\mathrm{M},A,B}$ by (1). □

Exceptional points for the birational equivalence

Remark 5.13. The map $(u,v) \mapsto (u/v,(u-1)/(u+1))$ from $\mathrm{E}_{\mathrm{M},A,B}$ to $\mathrm{E}_{\mathrm{E},a,d}$ in Theorem 5.12 is undefined at the points of $\mathrm{E}_{\mathrm{M},A,B} : Bv^2 = u^3 + Au^2 + u$ with $v = 0$ or $u + 1 = 0$. We investigate these points in more detail:

(1) The point $(0,0)$ on $\mathrm{E}_{\mathrm{M},A,B}$ corresponds to the affine point of order 2 on $\mathrm{E}_{\mathrm{E},a,d}$, namely $(0,-1)$. This point and $(0,1)$ are the only exceptional points of the inverse map $(x,y) \mapsto ((1+y)/(1-y),(1+y)/(1-y)x)$, where $(0,1)$ is mapped to the point at infinity.

(2) If $(A+2)(A-2)$ is a square (i.e. if ad is a square) then there are two more points with $v = 0$, namely $((-A \pm \sqrt{(A+2)(A-2)})/2, 0)$. These points have order 2. These points correspond to two points of order 2 at infinity on the desingularisation of $\mathrm{E}_{\mathrm{E},a,d}$.

(3) If $(A-2)/B$ is a square (i.e. if d is a square) then there are two points with $u = -1$, namely $(-1, \pm\sqrt{(A-2)/B})$. These points have order 4. These points correspond to two points of order 4 at infinity on the desingularisation of $\mathrm{E}_{\mathrm{E},a,d}$.

5.3.2. Arithmetic on twisted Edwards curves

In this section we derive fast explicit formulas for addition and doubling on twisted Edwards curves.

Let $(x_1,y_1),(x_2,y_2)$ be points on (5.6). The addition law on twisted Edwards curves is given by

$$(x_1,y_1)+(x_2,y_2) = \left(\frac{x_1y_2+y_1x_2}{1+dx_1x_2y_1y_2}, \frac{y_1y_2-ax_1x_2}{1-dx_1x_2y_1y_2}\right).$$

The neutral element is $(0,1)$ and $-(x_1,y_1) = (-x_1,y_1)$.

These formulas work for addition as well as for doubling. The remark in the previous section considers points for which the birational map to a Montgomery curve is not defined. This implies that the desingularisation of a twisted Edwards curve has no points at infinity exactly if a is a square in k and d is a non-square in k. This implies that the addition law is complete exactly if the curve is isomorphic to a complete Edwards curve, i.e. an Edwards curve with a non-square d.

For the correctness of the addition law observe that it coincides with the Edwards addition law on $\bar{x}^2 + y^2 = 1 + (d/a)\bar{x}^2y^2$ with $\bar{x} = x\sqrt{a}$ which is proven correct in [BL07b, Section 3].

5.3.3. Arithmetic in projective form

To avoid inversions, we work on the projective twisted Edwards curve

$$(aX^2 + Y^2)Z^2 = Z^4 + dX^2Y^2.$$

For $Z_1 \neq 0$ the homogeneous point $(X_1 : Y_1 : Z_1)$ represents the affine point $(X_1/Z_1, Y_1/Z_1)$.

Addition

The following formulas compute an addition in 10M+1S+2D+7add.

$$\begin{aligned} A &= Z_1 \cdot Z_2;\ B = A^2;\ C = X_1 \cdot X_2;\ D = Y_1 \cdot Y_2;\ E = d \cdot C \cdot D; \\ F &= B - E;\ G = B + E;\ X_3 = A \cdot F \cdot ((X_1 + Y_1) \cdot (X_2 + Y_2) - C - D); \\ Y_3 &= A \cdot G \cdot (D - a \cdot C);\ Z_3 = F \cdot G. \end{aligned}$$

Remark 5.14. If a is a square in k, then the twisted Edwards curve $a\bar{x}^2 + \bar{y}^2 = 1 + d\bar{x}^2\bar{y}^2$ is isomorphic to the Edwards curve $x^2 + y^2 = 1 + (d/a)x^2y^2$, where $x = \bar{x}\sqrt{a}$ and $y = \bar{y}$ (cf. Section 5.3.5). One can perform additions on the Edwards curve with the following explicit formulas which need 10M+1S+3D+7add, where the 3D are two multiplications by a and one by d. The formulas for twisted Edwards curves are faster by 1 multiplication by a.

$$\begin{aligned} A &= Z_1 \cdot Z_2;\ B = a \cdot A^2;\ A = a \cdot A;\ C = X_1 \cdot X_2;\ D = Y_1 \cdot Y_2;\ E = d \cdot C \cdot D; \\ F &= B - E;\ G = B + E;\ X_3 = A \cdot F \cdot ((X_1 + Y_1) \cdot (X_2 + Y_2) - C - D); \\ Y_3 &= A \cdot G \cdot (D - C);\ Z_3 = F \cdot G. \end{aligned}$$

Doubling

The following formulas compute doubling in 3M+4S+1D+7add, where the 1D is the multiplication by a.

$$\begin{aligned} B &= (X_1 + Y_1)^2;\ C = X_1^2;\ D = Y_1^2;\ E = a \cdot C; F := E + D;\ H = Z_1^2; \\ J &= F - 2H;\ X_3 = c \cdot (B - C - D) \cdot J;\ Y_3 = F \cdot (E - D);\ Z_3 = F \cdot J. \end{aligned}$$

5.3.4. Arithmetic in inverted form

The generalisation of inverted Edwards coordinates (see Bernstein and Lange [BL07c]) to twisted Edwards curves lets a point $(X_1 : Y_1 : Z_1)$ with $X_1Y_1Z_1 \neq 0$ on the curve

$$(X^2 + aY^2)Z^2 = X^2Y^2 + dZ^4$$

correspond to the affine point $(Z_1/X_1, Z_1/Y_1)$.

Addition

The addition formulas in inverted coordinates on the twisted Edwards curve are:

$$\begin{aligned} A &= Z_1 \cdot Z_2;\ B = dA^2;\ C = X_1 \cdot X_2;\ D = Y_1 \cdot Y_2;\ E = C \cdot D; \\ H &= C - aD;\ I = (X_1 + Y_1) \cdot (X_2 + Y_2) - C - D; \\ X_3 &= (E + B) \cdot H;\ Y_3 = (E - B) \cdot I;\ Z_3 = A \cdot H \cdot I. \end{aligned}$$

One readily counts 9M+1S+2D+7add, where the 2D are one multiplication by a and one by d.

Doubling

The doubling formulas in inverted coordinates on the twisted Edwards curve are:

$$\begin{aligned} A &= X_1^2;\ B = Y_1^2;\ U = aB;\ C = A + U;\ D = A - U;\ E = (X_1 + Y_1)^2 - A - B; \\ X_3 &= C \cdot D;\ Y_3 = E \cdot (C - 2d \cdot Z_1^2);\ Z_3 = D \cdot E. \end{aligned}$$

The computation needs 3M+4S+2D+6add, where the 2D are one multiplication by a and one by d.

5.3.5. Twisted Edwards curves as twists of Edwards curves

The twisted Edwards curve $\mathrm{E}_{\mathrm{E},a,d} : ax^2 + y^2 = 1 + dx^2y^2$ is a quadratic twist of the Edwards curve $\mathrm{E}_{\mathrm{E},1,d/a} : \bar{x}^2 + \bar{y}^2 = 1 + (d/a)\bar{x}^2\bar{y}^2$. The map $(\bar{x}, \bar{y}) \mapsto (x, y) = (\bar{x}/\sqrt{a}, \bar{y})$ is an isomorphism from $\mathrm{E}_{\mathrm{E},1,d/a}$ to $\mathrm{E}_{\mathrm{E},a,d}$ over $k(\sqrt{a})$. If a is a square in k, then $\mathrm{E}_{\mathrm{E},a,d}$ is isomorphic to $\mathrm{E}_{\mathrm{E},1,d/a}$ over k.

More generally, $\mathrm{E}_{\mathrm{E},a,d}$ is a quadratic twist of $\mathrm{E}_{\mathrm{E},\bar{a},\bar{d}}$ for any $\bar{a}, \bar{d}$ satisfying $\bar{d}/\bar{a} = d/a$. Conversely, every quadratic twist of a twisted Edwards curve is isomorphic to a twisted Edwards curve, i.e. the set of twisted Edwards curves is invariant under quadratic twists.

Furthermore, the twisted Edwards curve $\mathrm{E}_{\mathrm{E},a,d} : ax^2 + y^2 = 1 + dx^2y^2$ is birationally equivalent to the twisted Edwards curve $\mathrm{E}_{\mathrm{E},d,a} : d\bar{x}^2 + \bar{y}^2 = 1 + a\bar{x}^2\bar{y}^2$. The map $(\bar{x}, \bar{y}) \mapsto (x, y) = (\bar{x}, 1/\bar{y})$ is a birational equivalence from $\mathrm{E}_{\mathrm{E},d,a}$ to

$\mathrm{E}_{\mathrm{E},a,d}$. More generally, $\mathrm{E}_{\mathrm{E},a,d}$ is a quadratic twist of $\mathrm{E}_{\mathrm{E},\bar{a},\bar{d}}$ for any $\bar{a}, \bar{d}$ satisfying $\bar{d}/\bar{a} = a/d$. This generalises the known fact, used in [BL07b, proof of Theorem 2.1], that $\mathrm{E}_{\mathrm{E},1,d}$ is a quadratic twist of $\mathrm{E}_{\mathrm{E},1,1/d}$.

5.4. Doubling and tripling on Edwards curves

As we have already seen in Chapters 3 and 4, doubling formulas are essential for the computation of scalar multiples, e.g. when we use the double-and-add algorithm or a variant of it.

Another approach is to use the double-base number system ([DIM05, DI06]), where the scalar is represented by a sum of powers of two base elements; in this case we speak of a double-base chain (cf. Section 5.5.1). Very often, the basis is $\{2, 3\}$ and hence we need efficient doubling and tripling formulas.

In the following, we provide doubling and tripling formulas (see also Section 2 in [BBLP07] and Section 4 in [BL07b]) for Edwards curves. We obtained the doubling formulas from the addition formulas (5.3) with $x_1 = x_2$ and $y_1 = y_2$, and we found the tripling formulas by composing the addition and the doubling formulas. All of them have been verified with the help of Magma [BCP97].

The double of a point (x_1, y_1) on the Edwards curve given by the equation $x^2 + y^2 = 1 + dx^2y^2$ is

$$[2](x_1, y_1) = \left(\frac{2x_1y_1}{x_1^2 + y_1^2}, \frac{y_1^2 - x_1^2}{2 - (x_1^2 + y_1^2)}\right), \tag{5.8}$$

while the triple is

$$[3](x_1, y_1) = \left(\frac{(x_1^2 + y_1^2)^2 - (2y_1)^2}{4(x_1^2 - 1)x_1^2 - (x_1^2 - y_1^2)^2}x_1, \frac{(x_1^2 + y_1^2)^2 - (2x_1)^2}{-4(y_1^2 - 1)y_1^2 + (x_1^2 - y_1^2)^2}y_1\right). \tag{5.9}$$

Another interesting use of the doubling and tripling formulas is the generation of Edwards curves over $\mathbb{Q}$ with prescribed torsion subgroup. In Section 6.2.1 we use doubling formulas to generate a point of order 8 which allows us to generate an Edwards curve with torsion subgroup isomorphic to $\mathbb{Z}/2\mathbb{Z}\times\mathbb{Z}/8\mathbb{Z}$. In Section 6.2.2 we show how to generate an Edwards curve with torsion subgroup isomorphic to $\mathbb{Z}12$ by finding a point of order 3 using the tripling formulas.

5.5. Impact of point tripling on double-base chains

In this section, we investigate the influence of triplings on double-base chains for many different shapes of elliptic curves and many different coordinate systems. In particular, we look at the relevance of double-base chains for scalar multiplication

on Edwards curves. The main result of this section is that double-base chains offer no advantage for scalar multiplication on Edwards curves.

In the following, we explain single-base and double-base chains, describe the different curve shapes that we used in our experiments and give details about the parameters of the experiments. This section is based on joint work with Bernstein, Lange and Peters [BBLP07].

5.5.1. Double-base chains

A *single-base chain* is an expansion of the scalar multiplication $[n]P$ as

$$[n]P = \sum c_i 2^i P \quad \text{with } c_i \in \{0, 1\}, \tag{5.10}$$

where the basis is the set $\{2\}$.

In 2005, *double-base chains* were introduced by Dimitrov, Imbert and Mishra [DIM05]. The main idea of double-base chains is to expand the scalar multiplication $[n]P$ in various ways as

$$[n]P = \sum c_i 2^{a_i} 3^{b_i} P \quad \text{with } c_i \in \{-1, 1\}. \tag{5.11}$$

This can be seen as a generalisation of single-base chains with basis $\{2, 3\}$.

There were several previous "double-base number system" results expanding $[n]P$ in that way, e.g. using greedy algorithms, but they are mostly quite expensive, which makes this approach unsuitable in practise. The critical advance in [DIM05] was to require $a_1 \geq a_2 \geq a_3 \geq \ldots$ and $b_1 \geq b_2 \geq b_3 \geq \ldots$, allowing a straightforward computation without the expensive backtracking that plagued previous papers. In this paper, the authors use a Horner-like evaluation, so the representation is "easy" to compute since only a_1 doublings and b_1 triplings need to be computed. However, this approach comes at the cost of more additions.

A further improvement was presented by Doche and Imbert [DI06] in 2006. The authors suggest to use sliding windows and to choose $\pm c_i$ from a set S that is larger than the usual choice $S = \{1\}$ in [DIM05]. For the choices of S in our experiments see Section 5.5.3.

5.5.2. Curves shapes and coordinate systems

For our experiments we have taken many different shapes of elliptic curves and many different coordinate systems into account. In the following, we give a brief explanation of the shapes we used. In this section, let E be an elliptic curve over a field k of characteristic at least 5.

The most popular representation of an affine point is in Jacobian coordinates. A point (x_1, y_1) on E is represented as $(X_1 : Y_1 : Z_1)$, satisfying the equation $Y_1^2 = X_1^3 + a_4 X_1 Z_1^2 + a_6 Z_1^6$. An addition of generic points $(X_1 : Y_1 : Z_1)$ and

$(X_2 : Y_2 : Z_2)$ in Jacobian coordinates costs 11M+5S. A doubling, i.e. an addition, where $(X_1 : Y_1 : Z_1)$ and $(X_2 : Y_2 : Z_2)$ are equal, costs 1M+8S. A tripling costs 5M+10S.

If $a_4 = -3$, then the cost for doubling changes to 3M+5S and that for tripling to 7M+7S. Not every curve can be transformed to allow $a_4 = -3$, but important examples such as the NIST curves [P1300] make this choice. We refer to this case as Jacobian-3.

Most of the literature presents slower formulas producing the same output, and correspondingly reports higher costs for arithmetic in Jacobian coordinates. See, for example, the P1363 standards [P1300] and the aforementioned overviews. We include the slower formulas in our experiments to simplify the comparison of our results to previous results in [DI06] and [DIM05]. We refer to the slower formulas as Std-Jac and Std-Jac-3.

"ExtJQuartic" and "Hessian" and "JacIntersect" refer to the latest addition formulas for Jacobi quartics $Y^2 = X^4 + 2aX^2Z^2 + Z^4$, Hessian curves $X^3 + Y^3 + Z^3 = 3dXYZ$, and Jacobi intersections $S^2 + C^2 = T^2, aS^2 + D^2 = T^2$.

"3DIK" is an abbreviation for "tripling-oriented Doche-Icart-Kohel curves," the curves $Y^2 = X^3 + a(X + Z^2)^2Z^2$ introduced in 2006 [DIK06].

5.5.3. Results

Figure 5.1 gives an overview of the results of our experiments, which included several bit sizes ℓ, namely 160, 200, 256, 300, 400, and 500. The choices 200, 300, 400, 500 were used in [DI06], and we include them to ease comparison. The choices 160 and 256 are common in cryptographic applications. Our experiments also included many choices of the parameter a_0 in [DI06, Algorithm 1]. The largest power of 2 allowed in the algorithm is 2^{a_0}, and the largest power of 3 allowed in the algorithm is 3^{b_0}, where $b_0 = \lceil (\ell - a_0)/\lg 3 \rceil$. Specifically, we tried each $a_0 \in \{0, 10, 20, \ldots, 10\lfloor \ell/10 \rfloor\}$. This matches the experiments reported in [DI06] for $\ell = 200$.

Our experiments included several coefficient sets S, i.e. sets of coefficients c allowed in $c2^a3^b$:

- the set $\{1\}$ used in [DIM05];
- the sets $\{1, 5, 7, 11, 13, 17, 19, 23, 25\}$, $\{1, 2, 3, 4, 8, 9, 16, 27, 81\}$, $\{1, 5, 7\}$ and $\{1, 2, 3\}$ appearing in the graphs in [DI06, Appendix B] with labels "$(1, 1)$" and "$(4, 4)$" and "S_2" and "S_8";
- the sets $\{1, 2, 3, 4, 9\}$, $\{1, 2, 3, 4, 8, 9, 27\}$, $\{1, 5\}$, $\{1, 5, 7, 11\}$, $\{1, 5, 7, 11, 13\}$, $\{1, 5, 7, 11, 13, 17, 19\}$ appearing in the tables in [DI06, Appendix B].

We also included the sets $\{1, 2, 3, 5\}$, $\{1, 2, 3, 5, 7\}$, $\{1, 2, 3, 5, 7, 9\}$ and so on through $\{1, 2, 3, 5, 7, 9, 11, 13, 15, 17, 19, 23, 25\}$. These sets are standard in the

base-2 context, but do not seem to have been included in previous double-base experiments. We used straightforward combinations of additions, doublings and triplings for the initial computation of cP for each $c \in S$.

We follow the standard (although debatable) practise of counting S=0.8M and disregarding other field operations.

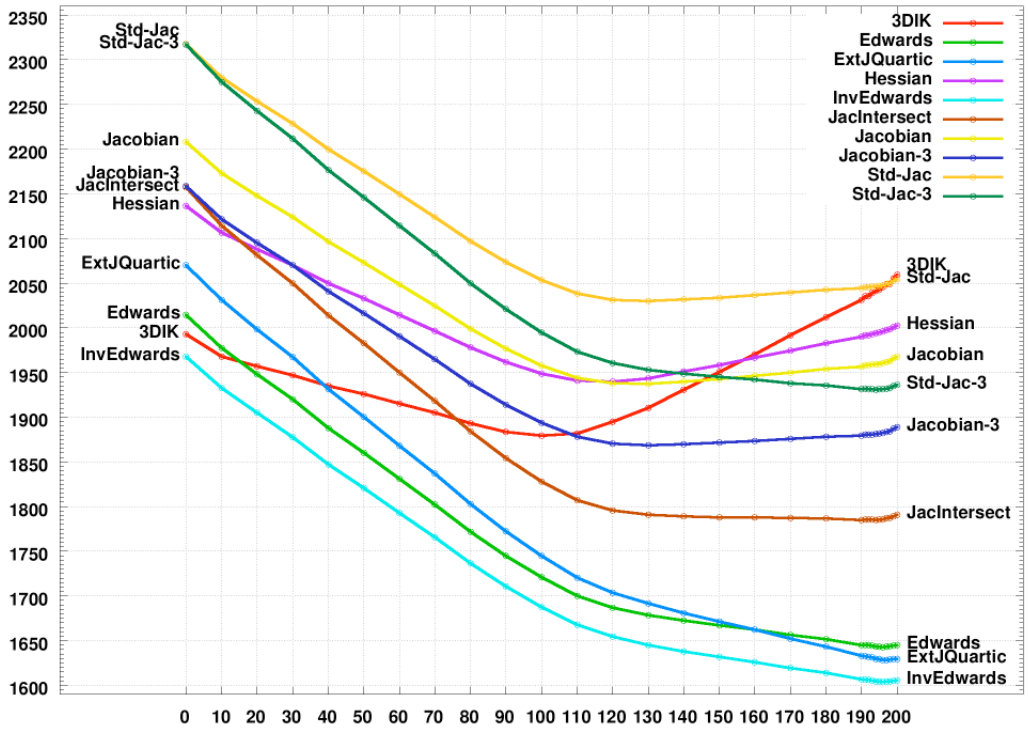


Figure 5.1.: Doubling/tripling ratios for double-base chains in various coordinate systems

There are 8236 combinations of ℓ, a_0 and S described above. For each combination, we

- generated 10000 uniform random integers $n \in \{0, 1, \ldots, 2^\ell - 1\}$,
- converted each integer into a chain as specified by a_0 and S,
- checked that the chain indeed computed n starting the chain from 1 and
- counted the number of triplings, doublings, additions, re-additions and mixed additions for those 10000 choices of n.

We converted the results into multiplication counts for ExtJQuartic, 3DIK, Hessian, Edwards, InvEdwards, JacIntersect, Jacobian, Jacobian-3, Std-Jac and

Std-Jac-3, obtaining a cost for each of the 82360 combinations of ℓ, curve shape, a_0 and S.

Figure 5.1 shows, for each a_0 (horizontal axis) and each curve shape, the cost (in multiplications) for $\ell = 200$ when S is chosen optimally. This graph demonstrates the importance of choosing the right bounds for a_0 and b_0 depending on the ratio of the doubling/tripling costs.

Curve shape	Mults	a_0	a_0/ℓ	S
3DIK	1879.200960	100	0.5	$\{1, 2, 3, 5, 7\}$
Edwards	1642.867360	196	0.98	$\{1, 2, 3, 5, \ldots, 15\}$
ExtJQuartic	1628.386660	196	0.98	$\{1, 2, 3, 5, \ldots, 15\}$
Hessian	1939.682780	120	0.6	$\{1, 2, 3, 5, \ldots, 13\}$
InvEdwards	1603.737760	196	0.98	$\{1, 2, 3, 5, \ldots, 15\}$
JacIntersect	1784.742	190	0.95	$\{1, 2, 3, 5, \ldots, 15\}$
Jacobian	1937.129960	130	0.65	$\{1, 2, 3, 5, \ldots, 13\}$
Jacobian-3	1868.530560	130	0.65	$\{1, 2, 3, 5, \ldots, 13\}$

Table 5.1.: Multiplication counts and percentage of doublings for various coordinate systems

In Table 5.1, we show basically the same results as in Figure 5.1, but the impact of the doubling/tripling ratio can be seen more clearly. In the third column, we list the number of doublings in the double-base chain for an optimal choice of the set S. The fourth column shows the same number in percent of $\ell = 200$. The optimal choice of S is given in the last column. One can see that for the three systems Edwards, ExtJQuartic and InvEdwards the amount of doublings is almost 100%.

These three systems are also the fastest ones. They need the lowest number of multiplications for values of a_0 very close to ℓ. These systems are using larger sets of precomputations than slower systems such as Jacobian-3 or Jacobian and fewer triplings. The faster systems all come with particularly fast addition laws, making the precomputations less costly, and particularly fast doublings, making triplings less attractive. This means that currently double-base chains offer no or very little advantage for the fastest systems. See [BL07b] for a detailed description of single-base scalar multiplication on Edwards curves.

6. Edwards Curves over $\mathbb{Q}$ and Applications to Factoring

In this chapter, we use Edwards and twisted Edwards curves, which we discussed in detail in the preceding chapter, to improve the speed of the Elliptic Curve Method of factoring. The most expensive operation in ECM is scalar multiplication of points on an elliptic curve over the rational numbers. Since Edwards curves currently offer the best speed for scalar multiplication (when representing the points and performing the computations in inverted Edwards coordinates [BL07c]), we want to find Edwards and twisted Edwards curves which are suitable for ECM, i.e. curves over $\mathbb{Q}$ with large torsion subgroup and positive rank (see Chapter VIII, §10 in[Sil86]).

In Section 6.1, we give an overview of ECM. In the next section, we show how to construct Edwards curves over $\mathbb{Q}$ with large torsion subgroup. For elliptic curves in Weierstraß form there are two parametrisation that generate curves over $\mathbb{Q}$ with torsion subgroup consisting of 6 (Suyama) and 16 (Atkin and Morain) elements. We translate both parametrisations to Edwards form. Then we show how to find curves with large torsion subgroup, small-height coefficients and small-height parameters to speed up ECM.

The work in this chapter is based on joint work with Bernstein, Lange and Peters. A preliminary version was published as [BBLP08].

6.1. Lenstra's elliptic curve method

In 1987, H. W. Lenstra [Len87] proposed a method for factoring integers using elliptic curves, the elliptic curve method (short: ECM). The method consists of two parts which are called "Stage 1" and "Stage 2". In this chapter we give an overview of Stage 1 and show how to construct elliptic curves in Edwards form such that they are suitable for ECM.

Stage 1 of ECM tries to factor a positive integer n as follows. Choose an elliptic curve E defined over $\mathbb{Q}$. Choose a rational function $\varphi : E \to \mathbb{Q}$ that has a pole at the neutral element of E; for example choose φ as the Weierstraß x-coordinate. Choose a non-torsion element $P \in E(\mathbb{Q})$. Choose a positive integer s with many small prime factors. Choose a sequence of additions, subtractions, multiplications and divisions to compute $\varphi([s]P)$, where $[s]P$ denotes the sth multiple of P in $E(\mathbb{Q})$. Compute $\varphi([s]P)$ modulo n by carrying out this sequence

of additions, subtractions, multiplications and divisions modulo n. Hope for an impossible division modulo n. An attempt to divide by a non-zero non-unit modulo n immediately reveals a factor of n; an attempt to divide by 0 modulo n is not quite as informative but usually allows a factor of n to be obtained without much extra work.

If n has a prime divisor q such that $[s]P$ is the neutral element of $E(\mathbb{Z}/q\mathbb{Z})$ then the Stage-1 ECM computation will involve an impossible division modulo n, usually revealing a factor of n. This occurs, in particular, whenever s is a multiple of the group size $\#E(\mathbb{Z}/q\mathbb{Z})$. As E varies randomly, $\#E(\mathbb{Z}/q\mathbb{Z})$ varies randomly in the Hasse interval $[q - 2\sqrt{q} + 1, q + 2\sqrt{q} + 1]$. What makes ECM useful is that a surprisingly small s, allowing a surprisingly fast computation of $[s]P$, is a multiple of a surprisingly large percentage of the integers in the Hasse interval, and is a multiple of the order of P modulo q with (conjecturally) an even larger probability.

6.1.1. Example

For example, one could try to factor n as follows. Choose the curve $E : y^2 = x^3 - 2$, the Weierstraß x-coordinate as φ, the point $(x, y) = (3, 5)$ and the integer $s = 420 = 2^2 \cdot 3 \cdot 5 \cdot 7$. Choose the following strategy to compute the x-coordinate of $[420](3, 5)$: use the standard affine-coordinate doubling formulas to compute $[2](3, 5)$, then $[4](3, 5)$, then $[8](3, 5)$; use the standard affine-coordinate addition formulas to compute $[12](3, 5)$; continue similarly through $[2](3, 5)$, $[4](3, 5)$, $[8](3, 5)$, $[12](3, 5)$, $[24](3, 5)$, $[48](3, 5)$, $[96](3, 5)$, $[192](3, 5)$, $[384](3, 5)$, $[408](3, 5)$, $[420](3, 5)$. Carry out these computations modulo n, hoping for a division by a non-zero non-unit modulo n.

The denominator of the x-coordinate of $[420](3, 5)$ in $E(\mathbb{Q})$ has many small prime factors: 2, 3, 5, 7, 11, 19, 29, 31, 41, 43, 59, 67, 71, 83, 89, 109, 163, 179, 181, 211, 223, 241, 269, 283, 383, 409, 419, 433, 523, 739, 769, 811, 839, etc. If n shares any of these prime factors then the computation of $[420](3, 5)$ will encounter an impossible division modulo n. To verify the presence of (for example) the primes 769, 811, and 839 one can observe that $[420](3, 5)$ is the neutral element in each of the groups $E(\mathbb{Z}/769\mathbb{Z})$, $E(\mathbb{Z}/811\mathbb{Z})$, $E(\mathbb{Z}/839\mathbb{Z})$; the order of $(3, 5)$ turns out to be 7, 42, 35 respectively. Note that the group orders are 819, 756, and 840, none of which divide 420.

6.1.2. The standard choice of s

Pollard in [Pol78, page 527] suggested choosing s as "the product of all the primes $p_i \leq L$ each to some power $c_i \geq 1$. There is some freedom in the choice of the c_i but the smallest primes should certainly occur to some power higher than the first."

Pollard's prime bound "L" is now called B_1. One possibility is to choose, for each prime $\pi \leq B_1$, the largest power of π in the interval $[1, n + 2\sqrt{n} + 1]$. Then $[s]P$ is the neutral element in $E(\mathbb{Z}/q\mathbb{Z})$ if and only if the order of P is "B_1-smooth", i.e. if and only if the order has no prime divisors larger than B_1. This possibility is theoretically pleasing but clearly suboptimal.

Brent in [Bre86, Section 5] says that "in practise we choose" the largest power of π in the interval $[1, B_1]$ "because this significantly reduces the cost of a trial without significantly reducing the probability of success." GMP-ECM uses the same strategy; see [ZD07, page 529].

6.1.3. Speeding up ECM

In this section, we discuss how the elliptic curve method can be sped up. We suggest to use Edwards curves instead of Montgomery curves for two reasons: First, the scalar multiplication can be computed faster since the arithmetic on Edwards curves is more efficient than on Montgomery curves (which are traditionally used in ECM implementations). Second, we can construct Edwards curves over $\mathbb{Q}$ with large torsion group (in particular, larger than previous methods). Our implementation GMP-EECM (the additional letter "E" stands for "Edwards") [BBLP08] makes use of these improvements. We compare it to the widely used GMP-ECM implementation by Zimmermann et al. to demonstrate the advantages of Edwards curves for factoring with elliptic curves.

Edwards versus Montgomery arithmetic

The most time consuming operation in ECM is scalar multiplication $[s]P$ for a non-torsion point P and a scalar s as in the previous section. We now explain how this operation can be sped up when using Edwards arithmetic instead of Montgomery arithmetic.

In particular, the GMP-ECM implementation uses Montgomery coordinates for stage 1, with "PRAC," a particular differential addition chain introduced by Montgomery. Zimmermann and Dodson in [ZD07, page 532, Figure 2] report a total cost of 2193683 differential additions to multiply an elliptic-curve point by

$$2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot \ldots \cdot 999983 \approx 2^{1440508.1677}$$

in Montgomery coordinates. By adding a few counters to the GMP-ECM 6.1.3 source code, we observed that GMP-ECM's stage 1, with $B_1 = 10^6$ and hence $s \approx 2^{1442098.6271}$, used 12982280 multiplications modulo n for 2196070 elliptic-curve additions, of which only 194155 were doublings.

The speedups of the GMP-EECM implementation are as follows:

(1) GMP-EECM uses twisted Edwards curves $ax^2+y^2 = 1+dx^2y^2$ with inverted Edwards coordinates with $\varphi = 1/x$ whereas GMP-ECM uses Montgomery curves with Montgomery coordinates.

(2) GMP-EECM handles the prime factors π of s in batches, whereas GMP-ECM handles each prime factor separately. GMP-EECM computes the product t of a batch, replaces P with $[t]P$, and then moves on to the next batch. We do not insist on batching *all* of the primes together (although we have done this in all computations so far); the cost of the multiplications to compute t should be balanced against the time saved by larger t. Note, however, that for small P there is no reason that $[t]P$ should be small, so the advantage of a small base point holds for only the first batch.

(3) GMP-EECM uses "signed sliding window" addition chains. These chains compute $P \mapsto [t]P$ using only 1 doubling and ε additions for each bit of t. Here ε converges to 0 as t increases in length; this is why a larger t saves time. Note that these chains are not compatible with Montgomery coordinates; they are shorter than any differential addition chain can be.

GMP-EECM follows tradition in its choice of s. We have considered, but not yet analysed or implemented, other choices of s; in particular, we comment that allowing prime powers in the larger interval $[1, B_1^{1.5}]$ would have negligible extra cost.

To understand the potential speedup here one can simply count multiplications. GMP-ECM uses approximately 9 multiplications for each bit of s, see the example with $B_1 = 10^6$ above.

Doubling in Edwards coordinates uses only 7 multiplications; addition in Edwards coordinates uses 12 multiplications but occurs for only a fraction ε of the bits of s. The total multiplication count $7 + 12\varepsilon$ is below 9 for $\varepsilon < 1/6$.

Of course, reality is more complicated than a multiplication count. One disadvantage of Edwards coordinates is the cost of computing products of batches of prime factors of s. One advantage of Edwards coordinates is that a larger fraction of the multiplications are squarings and multiplications by curve constants. Using inverted Edwards coordinates on twisted Edwards curves (as in GMP-EECM) has many more multiplications by curve constants, but this is a good tradeoff when the parameters are small.

A numerical example

We provided $n = (5^{367}+1)/(2 \cdot 3 \cdot 73219364069)$ as input to GMP-ECM 6.1.3, with stage-1 bound $B_1 = 16384$, on an Intel Pentium M (6b8) running at 800MHz. Stage 1 used 210299 multiplications modulo n and consumed a total of 2448 milliseconds.

We then provided the same input to our new GMP-EECM software. We used the same stage-1 bound and the same s, but we used our new curve $x^2 + y^2 = 1 + 161^2 x^2 y^2 / 289^2$ (see Section 6.2.7) in inverted twisted Edwards coordinates, with width-6 signed sliding windows. Stage 1 used only 195111 multiplications modulo n, consumed only 2276 milliseconds, and printed the (previously known)

prime 70057995652034894429. We inspected the point order and found that it has largest prime factor 9103 and second-largest prime factor 2459.

Because GMP-EECM is very new we have not yet tried to use it to find record-setting factorisations.

Large torsion group

The success of ECM depends on finding a point with smooth order modulo prime numbers (of good reduction). More precisely, we want to find a non-torsion point P on an elliptic curve E over $\mathbb{Q}$ such that this point has smooth order on E modulo a prime number p of good reduction. If the curve E over $\mathbb{Q}$ has a torsion subgroup with m elements, then we know that the reduced curve E modulo p has group order divisible by m. So the reduction guarantees a factor of m in the order of E modulo p. The larger this guaranteed factor is, the higher is the chance of finding a point with smooth order on E modulo p. The group order of elliptic curves over finite fields is investigated in [How93], the distribution of the group order is studied in [McK99].

In Sections 6.2.1 and 6.2.2, we show how to construct Edwards curves over $\mathbb{Q}$ with torsion group isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$ and $\mathbb{Z}/12\mathbb{Z}$. With this, we can guarantee a factor of 12 and 16 in the group order of the Edwards curves modulo prime numbers. The widely known GMP-ECM implementation by Zimmermann et al. uses Montgomery curves with torsion groups of order 6 and can therefore guarantee a factor of 6 only.

6.2. Edwards curves suitable for ECM

The theorem of Mazur [Sil86, Theorem 7.5, page 223] says that the torsion group $E_{\text{tor}}(\mathbb{Q})$ of any elliptic curve E over $\mathbb{Q}$ is isomorphic to one of the following fifteen finite groups:

$$E_{\text{tor}}(\mathbb{Q}) \cong \begin{cases} \mathbb{Z}/m\mathbb{Z}, & m = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, \\ \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2m\mathbb{Z}, & m = 1, 2, 3, 4. \end{cases}.$$

Any elliptic curve in Edwards form has a point of order 4 as we have seen in the preceding chapter. It follows that the torsion group of an Edwards curve is isomorphic to either $\mathbb{Z}/4\mathbb{Z}$, $\mathbb{Z}/8\mathbb{Z}$, $\mathbb{Z}/12\mathbb{Z}$, $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$, or $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$.

The most interesting cases for ECM are $\mathbb{Z}/12\mathbb{Z}$ and $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$, since they force the group orders of E modulo primes p (of good reduction) to be divisible by 12 and 16, respectively. In this section, we show which conditions an Edwards curve $x^2 + y^2 = 1 + dx^2y^2$ over $\mathbb{Q}$ must satisfy to have torsion group isomorphic to $\mathbb{Z}/12\mathbb{Z}$ or $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$. We give parametrisations for both cases.

6.2.1. Torsion group $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$

Theorem 6.1 states a genus-0 cover of the set of Edwards curves over $\mathbb{Q}$ with torsion group $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$, i.e. a surjective map from a genus-0 curve to the set of all Edwards curves over $\mathbb{Q}$ with this torsion group. Theorem 6.3 states a rational cover (i.e. there exists a rational parametrisation that allows us to hit all Edwards curves over $\mathbb{Q}$ with torsion group $\mathbb{Z}/2\mathbb{Z}\times\mathbb{Z}/8\mathbb{Z}$) and identifies the degree of the cover. Theorem 6.2 identifies all the points of order 8 on such curves.

Theorem 6.1. The torsion group of an Edwards curve $x^2 + y^2 = 1 + dx^2y^2$ over $\mathbb{Q}$ is isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$ if and only if d is a square and there exists a rational number $x_8 \notin \{0, \pm 1\}$ satisfying $(2x_8^2 - 1)/x_8^4 = d$.

Proof. Assume that the torsion group is isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$. The point $(1, 0)$ has order 4, so there must be a point (x_8, y_8) on the curve with $[2](x_8, y_8) = (1, 0)$. This implies $y_8^2 - x_8^2 = 0$ by Formula (5.8), and then the curve equation $x_8^2+y_8^2 = 1+dx_8^2y_8^2$ implies $2x_8^2 = 1+dx_8^4$. In particular, $x_8 \notin \{0, \pm 1\}$ since $d \neq 1$, and therefore $d = (2x_8^2 - 1)/x_8^4$. Furthermore, the torsion group has three points of order 2 and so d must be a square.

Conversely, assume that d is a square and that $d = (2x_8^2 - 1)/x_8^4$. Then the curve (after desingularisation) has three points of order 2, and it also has the point (x_8, x_8) of order 8. The torsion group thus contains a copy of $\mathbb{Z}/2\mathbb{Z}\times\mathbb{Z}/8\mathbb{Z}$. By Mazur's theorem the torsion group cannot be larger. □

Theorem 6.2. Assume that $d \in \mathbb{Q} \setminus \{0, 1\}$ is a square, and that $x_8 \in \mathbb{Q} \setminus \{0, \pm 1\}$ satisfies $(2x_8^2 - 1)/x_8^4 = d$. Then the set of 8 points

$$\{(\pm x_8, \pm x_8)\,, \left(\pm 1/(x_8\sqrt{d}), \pm 1/(x_8\sqrt{d})\right)\},$$

where the signs are taken independently, is exactly the set of points of order 8 on the Edwards curve $x^2 + y^2 = 1 + dx^2y^2$ over $\mathbb{Q}$.

Proof. We will show that these 8 points are distinct points of order 8 on the curve. The torsion group of the curve is isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$ by Theorem 6.1, so it has exactly 8 elements of order 8, which must be exactly these points.

To see that the 8 points are distinct,suppose that $x_8 = \pm 1/(x_8\sqrt{d})$. Then $x_8^2\sqrt{d} = \pm 1$ so $dx_8^4 = 1$ so $2x_8^2 = 2$ so $x_8^2 = 1$, contradiction.

To see that the points $(\pm x_8, \pm x_8)$ are on the curve, use the equation $2x_8^2 - 1 = dx_8^4$. To see that the points $\left(\pm 1/(x_8\sqrt{d}), \pm 1/(x_8\sqrt{d})\right)$ are on the curve, observe that

$$1 + d\frac{1}{(\pm x_8\sqrt{d})^2}\frac{1}{(\pm x_8\sqrt{d})^2} = \frac{1 + dx_8^4}{dx_8^4} = \frac{2x_8^2}{dx_8^4} = \frac{2}{(\pm x_8\sqrt{d})^2},$$

again using the equation $2x_8^2 - 1 = dx_8^4$.

To see that all the points have order 8, observe that $[2](x_1, \pm x_1) = (\pm 1, 0)$ by Formula (5.8), and that $(\pm 1, 0)$ has order 4. □

Theorem 6.3. If $u \in \mathbb{Q} \setminus \{0, -1, -2\}$ then the Edwards curve $x^2+y^2 = 1+dx^2y^2$ over $\mathbb{Q}$, where

$$x_8 = \frac{u^2 + 2u + 2}{u^2 - 2}, \qquad d = \frac{2x_8^2 - 1}{x_8^4},$$

has $P_8 = (x_8, x_8)$ as a point of order 8 and has torsion group isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$.

Conversely, every Edwards curve over $\mathbb{Q}$ with torsion group isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$ is expressible in this way.

The parameters u, $2/u$, $-2(u+1)/(u+2)$, $-(2+u)/(1+u)$, $-(u+2)$, $-2/(u+2)$, $-u/(u+1)$, and $-2(u+1)/u$ give the same value of d and they are the only values giving this d.

Proof. By Theorem 6.1 the necessary and sufficient condition for $E_{\mathrm{tor}}(\mathbb{Q}) \cong \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$ is that there exists $x_8 \notin \{0, \pm 1\}$ satisfying $(2x_8^2 - 1)/x_8^4 = d$ and that d is a square, i.e. that $2x_8^2 - 1$ is a square.

The equation $2x_8^2 - 1 = r^2$ has 4 trivial solutions $(1, 1), (1, -1), (-1, 1)$, and $(-1, -1)$. These tuples violate the condition $x_8 \notin \{0, \pm 1\}$. There are no other solutions to $2x_8^2 - 1 = r^2$ that violate the condition on x_8.

We parametrise $r^2 = 2x_8^2 - 1$ by intersecting it with lines through $(1, -1)$, i.e. the lines given by $r = ux_8 - u - 1$.

$$\begin{aligned} 0 &= (ux_8 - u - 1)^2 - 2x_8^2 + 1 = (u^2 - 2)x_8^2 - 2u(u + 1)x_8 + (u + 1)^2 + 1 \\ &= (u^2 - 2)(x_8 - 1)(x_8 - (u^2 + 2u + 2)/(u^2 - 2)). \end{aligned}$$

A new solution to $r^2 = 2x_8^2 - 1$ in terms of u is given by $(x_8, r) = ((u^2 + 2u^2 + 2)/(u^2 - 2), (u^2 + 4u + 2)/(u^2 - 2))$, where the value for r is computed using the line.

This parametrisation cannot find $(1, 1)$, but this solution is excluded anyway. The solutions $(1, -1)$, $(-1, 1)$, and $(-1, -1)$ are found for $u = -2$, $u = -1$, and $u = 0$, respectively. So $u \in \mathbb{Q} \setminus \{0, -1, -2\}$ gives a complete parametrisation of all Edwards curves with $E_{\mathrm{tor}}(\mathbb{Q}) \cong \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$.

The value of x_8 is invariant under the change $u \leftarrow -2(u + 1)/(u + 2)$ since

$$\frac{4(u + 1)^2 - 4(u + 1)(u + 2) + 2(u + 2)^2}{4(u + 1)^2 - 2(u + 2)^2} = \frac{2u^2 + 4u + 4}{2u^2 - 4} = \frac{u^2 + 2u + 2}{u^2 - 2}.$$

The change $u \leftarrow 2/u$ changes the sign of x_8:

$$\frac{4 + 4u + 2u^2}{4 - 2u^2} = -\frac{u^2 + 2u + 2}{u^2 - 2}.$$

Since d depends only on x_8 these 2 transformations account for the first 4 values of u giving the same curve.

By Theorem 6.2 the same d, and thus the same curve, is obtained also for $x_8' = 1/(\sqrt{d}x_8) = x_8/r = \frac{u^2+2u+2}{u^2+4u+2}$. This value is obtained for the parameter $-(u+2)$ since

$$\frac{(u+2)^2 - 2(u+2) + 2}{(u+2)^2 - 2} = \frac{u^2+2u+2}{u^2+4u+2}.$$

The other 3 values are obtained by applying the same 2 transformations as before. Since there are exactly 8 points of order 8 and since each of them determines the curve, these are precisely the values of u leading to the same curve. □

6.2.2. Torsion group $\mathbb{Z}/12\mathbb{Z}$

Theorem 6.4 states a genus-0 cover of the set of Edwards curves over $\mathbb{Q}$ with torsion group $\mathbb{Z}/12\mathbb{Z}$. Theorem 6.6 states a rational cover. Theorem 6.5 identifies all the points of order 12 on such curves.

Theorem 6.4. The torsion group of an Edwards curve $x^2 + y^2 = 1 + dx^2y^2$ over $\mathbb{Q}$ is isomorphic to $\mathbb{Z}/12\mathbb{Z}$ if and only if there exists a rational number $y_6 \notin \{-2, -1/2, 0, \pm 1\}$ satisfying $(2y_6+1)/(y_6^3(y_6+2)) = d$ and such that $-(y_6^2+2y_6)$ is a square.

Proof. Rational points of order 3 or 6 are exactly the points (x_6, y_6) for which the x-coordinate of $[3](x_6, y_6)$ is 0. By (5.9) these are exactly the points for which $(x_6^2 + y_6^2)^2 = (2y_6)^2$. If this holds for one (x_6, y_6) then also for $(\pm x_6, \pm y_6)$, where the signs are taken independently. Up to signs this means that $x_6^2 + y_6^2 = -2y_6$. If the curve has such a point then d must satisfy the equation $x_6^2 + y_6^2 = 1 + dx_6^2y_6^2$, i.e. $-2y_6 = 1 + d(-2y_6 - y_6^2)y_6^2$. For $y_6 \notin \{-2, -1/2, 0, \pm 1\}$ the value $d = (2y_6+1)/(y_6^3(y_6+2))$ is defined and not equal to 0 or 1.

For this d we get a point of order 3 or 6 exactly if $x_6^2 = -(y_6^2 + 2y_6)$ has a rational solution.

Since each Edwards curve has a point of order 4 the torsion group must contain a copy of $\mathbb{Z}/12\mathbb{Z}$; by Mazur's theorem the torsion group cannot be larger. □

Theorem 6.5. Let $x^2+y^2 = 1+dx^2y^2$ be an Edwards curve over $\mathbb{Q}$ with $E_{\text{tor}}(\mathbb{Q}) \cong \mathbb{Z}/12\mathbb{Z}$ and let $P_3 = (x_3, y_3)$ be a point of order 3 on the curve.

The 12 torsion points on the curve and their respective orders are as follows:

point	$(0,1)$	$(0,-1)$	$(\pm x_3, y_3)$	$(\pm 1, 0)$	$(\pm x_3, -y_3)$	$(\pm y_3, \pm x_3)$
order	1	2	3	4	6	12

Proof. The points of order 6 are obtained as $(\pm x_3, y_3) + (0, -1)$, the points of order 12 by adding $(\pm 1, 0)$ to the points of order 3 and 6. □

Theorem 6.6. If $u \in \mathbb{Q} \setminus \{0, \pm 1\}$ then the Edwards curve $x^2 + y^2 = 1 + dx^2y^2$ over $\mathbb{Q}$, where

$$x_3 = \frac{u^2-1}{u^2+1}, y_3 = -\frac{(u-1)^2}{u^2+1}, d = \frac{(u^2+1)^3(u^2-4u+1)}{(u-1)^6(u+1)^2}$$

has $P_3 = (x_3, y_3)$ as a point of order 3 and has torsion group isomorphic to $\mathbb{Z}/12\mathbb{Z}$.

Conversely, every Edwards curve over $\mathbb{Q}$ with torsion group isomorphic to $\mathbb{Z}/12\mathbb{Z}$ is expressible in this way.

The parameters u and $1/u$ give the same value of d.

Proof. The points of order 3 are determined by $[2](x_3, y_3) = (-x_3, y_3)$ and $x_3, y_3 \neq 0$. Solving this equation gives $x_3^2 + y_3^2 = -2y_3$, i.e. $x_3^2 + (y_3 + 1)^2 = 1$. Parametrisation of the unit circle $r^2 + s^2 = 1$ (with $r = x_3$ and $s = y_3 + 1$) yields $(r, s) = (((u^2-1)/(u^2+1), 2u/(u^2+1))$ and thus the point

$$\begin{aligned}(x_3, y_3) &= \left((u^2-1)/(u^2+1), 2u/(u^2+1) - 1\right) \\ &= \left((u^2-1)/(u^2+1), -(u-1)^2/(u^2+1)\right).\end{aligned}$$

The parametrisation does not find the solution $(r, s) = (1, 0)$, i.e. $(x_3, y_3) = (1, -1)$ which is not a point of order 3. Likewise, $(r, s) = (-1, 0)$ which is found for $u = 0$ does not lead to a point of order 3. The solutions $(r, s) = (0, \pm 1)$, obtained for $u = \pm 1$ lead to $x_3 = 0$ and are therefore excluded.

The value for d follows from Theorem 6.4. For $u \in \mathbb{Q} \setminus \{0, \pm 1\}$ the value of d is defined and not equal to 0 or 1.

The value of d is invariant under the change $u \leftarrow 1/u$ since

$$\frac{(1+u^2)^3(1-4u+u^2)}{(1-u)^6(1+u)^2} = \frac{(u^2+1)^3(u^2-4u+1)}{(u-1)^6(u+1)^2}.$$

□

Solving the equation $d(u') = d(u)$ for u' in terms of u over the rationals shows that $u \leftarrow 1/u$ is the only rational transformation leaving d invariant that works independently of u.

6.2.3. Torsion group $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/6\mathbb{Z}$

Theorem 6.7. There exists no twisted Edwards curve over $\mathbb{Q}$ with torsion subgroup isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/6\mathbb{Z}$.

Proof. A twisted Edwards curve is birationally equivalent to a Montgomery curve (see Section 5.3.1 and Section 3 in [BBJ+08]). So for simplicity we give the proof for curves in Montgomery form.

Let $A, B \in \mathbb{Q}$ and let $E_{\mathrm{M},A,B} : By^2 = x^3 + Ax^2 + x$ be an elliptic curve over $\mathbb{Q}$ in Montgomery form. We show that there is no point of order 3 on $E_{\mathrm{M},A,B}$ and therefore no point of order 6.

The right-hand side of the curve equation is equal to $(x - a_1)(x - a_2)x$, where a_1, a_2 are distinct non-zero rational numbers. Due to Vieta's formula we know that $A = -(a_1 + a_2)$ and $1 = a_1 a_2$. Hence, $a_2 = 1/a_1$.

The condition to have a point $P_3 = (x_3, y_3)$ of order 3 on the curve is $[2]P_3 = -P_3 = (x_3, -y_3)$. Using the addition law on elliptic curves we get the equality for the x-coordinate $x_3 = B\lambda^2 - 2x_3 - A$, where $\lambda = (3x_3^2 + 2Ax_3 + 1)/(2By)$. Solving the equation for A, we get

$$A = -\frac{3x_3^4 + 6x_3^2 - 1}{4x_3^3} = -(a_1 + 1/a_1).$$

This can be written as the curve $a_1(3x_3^4 + 6x_3^2 - 1) = (a_1^2 + 1)4x_3^3$. Changing the variable names from a_1 to V and x_3 to U, we get (after homogenisation) the projective curve

$$C : V(3U^4 + 6U^2Z^2 - Z^4) = (V^2 + Z^2)4U^3,$$

which has rank 0, genus 1 and exactly the eight points $(4/3 : 1 : 0)$, $(0 : 1 : 0)$, $(0 : 0 : 1)$, $(1 : 0 : 0)$, $(-1/3 : 1 : 1)$, $(1/3 : -1 : 1)$, $(1 : 1 : 1)$ and $(-1 : -1 : 1)$. Three of the points are at infinity; the other five points give values for x_3 and A such that the appropriate Montgomery curve is singular. □

6.2.4. Parametrisations

Atkin and Morain in [AM93] found an infinite family of elliptic curves over $\mathbb{Q}$ with torsion group $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$ and with explicit non-torsion points.

In this section, we quote the Atkin-Morain theorem and then translate it from Weierstraß form to Edwards form.

Suyama in [Suy85] gives an infinite sequence of Montgomery curves which have torsion subgroup $\mathbb{Z}/6\mathbb{Z}$ over $\mathbb{Q}$. Note that these curves have group order divisible by 4 when considered modulo any odd prime number. We will convert this sequence of curves to twisted Edwards curves and discuss its use for ECM with Edwards curves.

6.2.5. Atkin and Morain's parametrisation

The Atkin-Morain family is parametrised by points (s, t) on a particular elliptic curve $T^2 = S^3 - 8S - 32$. Atkin and Morain suggest computing multiples (s, t) of $(12, 40)$, a non-torsion point on this curve. Beware that these points have rapidly increasing height.

Theorem 6.8 (Atkin, Morain)**.** Let (s,t) be a rational point on the curve $T^2 = S^3 - 8S - 32$. Define $\alpha = ((t+25)/(s-9)+1)^{-1}$, $\beta = 2\alpha(4\alpha+1)/(8\alpha^2-1)$, $c = (2\beta-1)(\beta-1)/\beta$, and $b = \beta c$. Then the elliptic curve

$$E_\alpha : Y^2 = X^3 + \frac{((c-1)^2-4b)}{4}X^2 + \frac{b(c-1)}{2}X + \frac{b^2}{4}$$

has torsion group isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$ and a non-torsion point with x-coordinate $-(2\beta-1)/4$.

Theorem 6.9. Let (s,t) be a rational point on the curve $T^2 = S^3 - 8S - 32$. Define α and β as in Theorem 6.8. Define $d = (2(2\beta-1)^2-1)/(2\beta-1)^4$. Then the Edwards curve $x^2 + y^2 = 1 + dx^2y^2$ has torsion group isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$ and a point (x_1, y_1) with $x_1 = (2\beta-1)(4\beta-3)/(6\beta-5)$ and $y_1 = (2\beta-1)(t^2+50t-2s^3+27s^2-104)/(t+3s-2)(t+s+16)$.

Proof. By construction $x_8 = 2\beta - 1$ satisfies $(2x_8^2-1)/x_8^4 = d$. Furthermore

$$d = \frac{(8\alpha^2-1)^2(8\alpha^2+8\alpha+1)^2}{(8\alpha^2+4\alpha+1)^4},$$

so d is a square. By Theorem 6.1, the Edwards curve has torsion group isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$. Finally, a straightforward calculation shows that $x_1^2 + y_1^2 = 1 + dx_1^2y_1^2$. □

The point with x-coordinate $-(2\beta-1)/4$ in Theorem 6.8 is generically a non-torsion point. The y-coordinate of the point is not stated explicitly in [AM93]. The point (x_1, y_1) in Theorem 6.9 is the corresponding point on the Edwards curve.

6.2.6. Suyama's parametrisation

The GMP-ECM package uses a family of elliptic curves in Montgomery form given by Suyama's parametrisation (see [ZD07]). We briefly revise this parametrisation and show how we can obtain a similar result for twisted Edwards curves.

Theorem 6.10. Let $\sigma > 5$ be an integer. We define

$$\begin{aligned} \alpha &= \sigma^2 - 5, \quad \beta = 4\sigma, \quad U_0 = \alpha^3, \quad W_0 = \beta^3, \\ A &= (\beta-\alpha)^3(3\alpha+\beta)/(4\alpha^3\beta) - 2, \quad B = \alpha/W_0. \end{aligned}$$

Then the elliptic curve $E_{M,A,B} : Bv^2 = u^3 + Au^2 + u$ has torsion group $\mathbb{Z}/6\mathbb{Z}$ over the rational numbers.

Let $V_0 = (\sigma^2-1)(\sigma^2-25)(\sigma^4-25)$. The point $(u_0, v_0) = (U_0/W_0, V_0/W_0)$ on $E_{M,A,B}$ is a non-torsion point.

Remark 6.11. It is well known that reducing the curve modulo a prime p yields a factor 4 in the group order of $E_{M,A,B}$. Thus, 12 divides $\#E_{M,A,B}(\mathbb{Z}/p\mathbb{Z})$.

Theorem 6.12. Let $\sigma > 5$ and $\alpha, \beta, U_0, V_0, W_0$ as in Theorem 6.10. For $a = (\beta-\alpha)^3(3\alpha+\beta)\beta^2/(4\alpha^4)$ and $d = (\beta+\alpha)^3(\beta-3\alpha)\beta^2/(4\alpha^4)$ the twisted Edwards curve $ax^2+y^2 = 1+dx^2y^2$ has group order divisible by 12 over any field of prime order and a point $(x_0, y_0) = (\alpha^3/V_0, (\alpha^3-\beta^3)/(\alpha^3+\beta^3))$.

Proof. We showed in Section 5.3.1 that over a non-binary field k every Montgomery curve $E_{M,A,B} : Bv^2 = u^3 + Au^2 + u$ is birationally equivalent to a twisted Edwards curve $E_{E,a,d} : ax^2 + y^2 = 1 + dx^2y^2$. The relations between the curve coefficients are $a = (A+2)/B$ and $d = (A-2)/B$ and the map from $E_{M,A,B}$ to $E_{E,a,d}$ is given by $(u, v) \mapsto (x, y) = (u/v, (u-1)/(u+1))$. With $A = (\beta-\alpha)^3(3\alpha+\beta)/(4\alpha^3\beta) - 2$ and $B = \alpha/\beta^3$ as in Theorem 6.10 we get the desired values for a and d. Mapping the point $(u_0, v_0) = (\alpha^3/\beta^3, V_0/\beta^3)$ to $E_{E,a,d}$ yields the desired point (x_0, y_0):

$$x_0 = u_0/v_0 = \alpha^3/V_0 \quad \text{and} \quad y_0 = \frac{u_0-1}{u_0+1} = \frac{\alpha^3-\beta^3}{\alpha^3+\beta^3}.$$

The theorem now follows from Remark 6.11. □

In order to compare the curves used in GMP-ECM to Edwards curves we considered Suyama's parametrisation for twisted Edwards curves. However, it is questionable if this parametrisation yields better results for GMP-EECM than the conditions on Edwards curves given in the previous section.

In our tests we considered Edwards curves with torsion group $\mathbb{Z}/12\mathbb{Z}$ as in Section 6.2 and computed the group order modulo primes p in the interval $[10^6, 2 \cdot 10^6]$. The average exponent of a factor 2 in the group order was 11/3 and the average exponent of a factor 3 was 5/3.

For Montgomery curves given by the Suyama parametrisation we get the same values for the exponents of 2 and 3 only in the case $\sigma = 11$ whereas for all other tested values of σ the group order has less small factors.

6.2.7. Edwards curves with small parameters

One way to save time in computations on generalised Edwards curves is to choose small parameters a, d and small points $(X_1 : Y_1 : Z_1)$; see Section 5.3. Another way to save time is to construct curves of rank at least 1 with large torsion over $\mathbb{Q}$. Unfortunately, essentially all of the curves constructed in the previous sections have large a, d, X_1, Y_1, Z_1.

Our aim in this section is to combine these two time-saving techniques, finding twisted Edwards curves that simultaneously have small parameters a, d, a small non-torsion point $(X_1 : Y_1 : Z_1)$, and large torsion over $\mathbb{Q}$.

Overall we found more than 100 small Edwards curves having small non-torsion points and at least 12 torsion points over $\mathbb{Q}$. Of course, one can easily write down many more small curves if one is willing to sacrifice some torsion.

Torsion group $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$

First we consider the case where the curve has torsion group isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$, i.e. there exists a point of order 8 on the curve and d is a square.

Theorem 6.2 states all points of order 8. The other affine points of finite order are $(0, \pm 1)$ and $(\pm 1, 0)$. Any other point (x_1, y_1) on the curve must have infinite order.

Theorem 6.3 gives a complete parametrisation of all curves with torsion group isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$. Any rational point (u, x_8, d, x_1, y_1) on the surface described by $x_8 = (u^2+2u+2)/(u^2-2)$, $d = (2x_8^2-1)/x_8^4$, and $x_1^2+y_1^2 = 1+dx_1^2y_1^2$ for $u \in \mathbb{Q} \setminus \{0, -1, -2\}$ gives us a suitable curve for ECM as long as we can ensure that (x_1, y_1) is none of the points of finite order.

We consider only $u > \sqrt{2}$. This does not cause any loss of generality: if $0 < u < \sqrt{2}$ then $2/u > \sqrt{2}$, and $2/u$ produces the same curve by Theorem 6.3; if $u < -2$ then $-(u+2) > 0$, and $-(u+2)$ produces the same curve by Theorem 6.3; if $-2 < u < -1$ then $-2(u+1)/(u+2) > 0$, and $-2(u+1)/(u+2)$ produces the same curve by Theorem 6.3; if $-1 < u < 0$ then $-u/(u+1) > 0$, and $-u/(u+1)$ produces the same curve by Theorem 6.3.

Write u as a/b for positive integers a, b. Expressing x_8 and d in terms of a and b produces the denominator $(a^2 + 2ab + 2b^2)^4$ for d and thus for $dx_1^2y_1^2$. Thus we scale x_1 and y_1 as

$$x_1 = (a^2 + 2ab + 2b^2)/e, y_1 = (a^2 + 2ab + 2b^2)/f.$$

Expressing all variables in a, b, e, f we find that solutions (u, x_8, d, x_1, y_1) correspond to integer solutions a, b, e, f of the $(1, 1, 2, 2)$-weighted-homogeneous equation

$$(e^2 - (a^2 + 2ab + 2b^2)^2)(f^2 - (a^2 + 2ab + 2b^2)^2) = (4ab(a + b)(a + 2b))^2.$$

We found many small solutions to this equation, and thus many of the desired Edwards curves, as follows. We considered a range of positive integers a. For each a we enumerated integers b between 1 and $\lfloor a/\sqrt{2} \rfloor$. For each (a, b) we enumerated all divisors of $(4ab(a + b)(a + 2b))^2$, added $(a^2 + 2ab + 2b^2)^2$ to each divisor, and searched for squares.

After about a week of computation on some computers at LORIA, roughly $2 \cdot 10^{16}$ CPU cycles in total, we had inspected more than 10^{14} divisors, found 25 different values of d, and checked that we had 25 different j-invariants.

Here are two examples:

(1) The solution $(a, b, e, f) = (3, 1, 19, 33)$ produces the order-8 point $(17/7, 17/7)$ and the non-torsion point $(17/19, 17/33)$ on the Edwards curve $x^2 + y^2 = 1 + dx^2y^2$ where $d = 161^2/17^4$.

(2) The solution $(a, b, e, f) = (24882, 9009, 258492663, 580153002)$ produces the non-torsion point $(86866/18259, 8481/4001)$ on the Edwards curve $x^2+y^2 = 1 + dx^2y^2$ where $d = 5657719^2/3341^4$.

The number of d's below height H appears to grow as roughly $\lg H$. For comparison, the Atkin-Morain procedure discussed in Section 6.2.5 generates only about $\sqrt{\lg H}$ examples below height H.

Torsion group $\mathbb{Z}/12\mathbb{Z}$

Writing $u = a/b$ in Theorem 6.6 yields an Edwards parameter d, a non-torsion point (x_1, y_1) and a point (x_3, y_3) of order 3 as follows:

$$\begin{aligned} d &= \frac{(a^2+b^2)^3(a^2-4ab+b^2)}{(a-b)^6(a+b)^2},\ x_3 = \frac{(a^2-b^2)}{(a^2+b^2)},\ y_3 = \frac{-(a-b)^2}{(a^2+b^2)}, \\ x_1 &= \frac{(a^2-b^2)}{e},\ y_1 = \frac{-(a-b)^2}{f}. \end{aligned}$$

We have to exclude x_1 from being any of the torsion points stated in Theorem 6.5 and need that $u \in \mathbb{Q} \setminus \{0, \pm 1\}$. Writing $x_1^2 + y_1^2 = 1 + dx_1^2y_1^2$ in terms of a, b, e, f shows that we have to look for points (a, b, e, f) on the surface

$$(e^2 - (a^2 - b^2)^2)(f^2 - (a - b)^4) = 16a^3b^3(a^2 - ab + b^2).$$

We found many small solutions as in Section 6.2.7: for each small (a, b) we enumerated all divisors of $16a^3b^3(a^2 - ab + b^2)$, added $(a^2 - b^2)^2$ to each divisor, and looked for squares.

After about a week of computation on some computers at LORIA we had found 78 different values of d and checked that we had 78 different j-invariants.

Here are two examples:

(1) The solution $(a, b, e, f) = (3, 2, 23, 7)$ produces the order-3 point $(5/13, -1/13)$ and the non-torsion point $(5/23, -1/7)$ on the Edwards curve $x^2 + y^2 = 1 + dx^2y^2$ where $d = -11 \cdot 13^3/5^2$.

(2) The solution $(a, b, e, f) = (15180, -7540, 265039550, 161866240)$ produces the non-torsion point $(3471616/5300791, -201640/63229)$ on the Edwards curve $x^2 + y^2 = 1 + dx^2y^2$ where $d = 931391 \cdot 359105^3/140003330048^2$.

7. Conclusions

In this thesis, we have studied two main topics. First, we investigated the use of hyperelliptic curves for cryptography. More precisely, we looked at hyperelliptic curves of genus 2 and 3 over binary fields, their Picard groups and especially the arithmetic on these groups. Second, we studied Edwards and twisted Edwards curves and how they can be applied for cryptography and for integer factorisation.

In Chapter 3 we studied hyperelliptic curves of genus 2 over finite fields of the form $\mathbb{F}_{2^d}$. As a starting point we have used the most general form of the curve equation in genus 2 and did a classification of all these curves according to their 2-rank. We found three main types of curves (and more sub types) and investigated the order of the Picard group (see Definition 2.14) of these curves for each class and gave explicit formulas for divisor class halving and divisor class doubling. It turned out that for some curve types the halving formulas require less field operations than the doubling ones. For other types the halving formulas can compete with their doubling counterparts, and for some types the halving formulas are slower. The main outcome is that halving formulas—together with a halve-and-add like algorithm—can improve the performance of scalar multiplication of divisor classes in the Picard group of genus-2 curves. This, in turn, can improve the overall speed of curve-based cryptosystems. In addition, in the second to last section of Chapter 3 we have given divisor class addition formulas and divisor class doubling formulas that are completely inversion-free. This is extremely important for applications in which field inversions are very costly (e.g. for hardware implementations).

We also answered the question on security at the very end of the chapter. We discussed the effects of currently known attacks to genus-2 curves with 2-rank 1. The result is that it is not difficult to select a curve such that the Picard group of this curve is secure against square-root attacks (Sections A.2 and A.3), Index calculus (Section A.3), MOV and Frey-Rück attacks (Section A.5) and Weil descent attacks (Section A.6). Hence we can recommend genus-2 curves for DLP-based cryptosystems since they can comply with the two most important criteria: efficient arithmetic and security.

In Chapter 4 we studied hyperelliptic curves of genus 3 over finite fields of the form $\mathbb{F}_{2^d}$. Analogously we did a classification of the curves, again, according to the 2-rank of the curves. It has turned out that curves with $h(x) = 1$ offer best performance for scalar multiplication in genus 3. In this case we could find explicit halving formulas that perform equally well as the doubling formulas. For curves of Type III (i.e. $h(x) = x$) we could develop halving formulas that are almost

twice as fast as the corresponding doubling ones, but curves of Type Ia are faster.

As in the genus-2 chapter we also checked the curves of genus 3 for susceptibility to currently known attacks and found that they can be suitable for cryptographic applications. Curves of higher genus are weak under Index Calculus attacks and cannot be recommended for DLP-based applications.

The third main subject in this thesis is Edwards curves (see Section 5) over fields of characteristic 0 or odd characteristic. We have explained Edwards and twisted Edwards curves and gave explicit formulas to perform the group law. It turned out that elliptic curves in Edwards form—when using Inverted Edwards coordinates—offer best speed for scalar multiplication. We have proved that all elliptic curves in Montgomery form are birationally equivalent to twisted Edwards curves (i.e. isomorphic with finitely many exceptions). This brought the speed of the Edwards addition law to all Montgomery curves.

We have also investigated the use of (twisted) Edwards curves over $\mathbb{Q}$ for the ECM factorisation method (cf. Section 6). We found parametrisations to generate infinitely many "good curves" for ECM, i.e. curves with large torsion subgroup, positive rank over $\mathbb{Q}$ and small-height parameters. First results showed a noticeable increase of the performance of ECM when using Edwards curves instead of elliptic curves in Weierstraß form. But not only the different form of the elliptic curve is responsible for the better performance, but also the small-height curve parameters and the larger torsion subgroup improved the integer factorisation.

7.1. Outlook

The halving formulas for curves of genus 2 and 3 work directly on the coefficients of the polynomials in the Mumford representation of a divisor class. For each case we have different explicit formulas. It would be interesting to investigate if it is possible to write down a halving algorithm that is completely general and works for any genus, like Cantor's algorithm does for divisor class doubling.

Edwards curves had such a large impact on elliptic-curve arithmetic that it is natural to ask if similar results can be obtained for higher-genus curves. A very interesting questions is whether it is possible to find an analogue of genus-2 curves in Edwards form, and if those can offer the same useful properties.

Bibliography

[ACD+05] Roberto M. Avanzi, Henri Cohen, Christophe Doche, Gerhard Frey, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. CRC Press, Boca Raton, 2005.

[AM93] A. O. L. Atkin and Francois Morain. Finding Suitable Curves for the Elliptic Curve Method of Factorization. *Mathematics of Computation*, 60(201):399–405, 1993.

[AT07] Roberto M. Avanzi and Nicolas Thériault. Effects of Optimizations for Software Implementations of Small Binary Field Arithmetic. In *International Workshop on the Arithmetic of Finite Fields – WAIFI 2007*, volume 4547 of *Lecture Notes in Computer Science*, pages 69–84. Springer-Verlag, 2007.

[ATW08] Roberto M. Avanzi, Nicolas Thériault, and Zheng Wang. Rethinking Low Genus Hyperelliptic Jacobian Arithmetic over Binary Fields: Interplay of Field Arithmetic and Explicit Formulæ. *Journal of Mathematical Cryptology*, 2:227–255, 2008.

[BBJ+08] Daniel J. Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted Edwards Curves. In *Progress in Cryptology – AFRICACRYPT 2008*, volume 5023 of *Lecture Notes in Computer Science*, pages 389–405. Springer-Verlag, 2008.

[BBLP07] Daniel J. Bernstein, Peter Birkner, Tanja Lange, and Christiane Peters. Optimizing double-base elliptic-curve single-scalar multiplication. In *Progress in Cryptology – INDOCRYPT 2007*, volume 4859 of *Lecture Notes in Computer Science*, pages 167–182. Springer-Verlag, 2007.

[BBLP08] Daniel J. Bernstein, Peter Birkner, Tanja Lange, and Christiane Peters. ECM using Edwards curves. Cryptology ePrint Archive of IACR, Report 2008/016, 2008. URL: `http://eprint.iacr.org/2008/016`.

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *Journal of Symbolic Computation*, 24(3–4):235–265, 1997.

[Be] Daniel J. Bernstein and Tanja Lange (editors). eBACS: ECRYPT Benchmarking of Cryptographic Systems. URL: `http://bench.cr.yp.to`. accessed 24 November 2008.

[Bir07] Peter Birkner. Efficient Divisor Class Halving on Genus Two Curves. In *Selected Areas in Cryptography*, volume 4356 of *Lecture Notes in Computer Science*, pages 317–326. Springer-Verlag, 2007.

[BL07a] Daniel J. Bernstein and Tanja Lange. Explicit-Formulas Database. URL: `http://www.hyperelliptic.org/EFD`, 2007.

[BL07b] Daniel J. Bernstein and Tanja Lange. Faster Addition and Doubling on Elliptic Curves. In *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 29–50. Springer-Verlag, 2007.

[BL07c] Daniel J. Bernstein and Tanja Lange. Inverted Edwards Coordinates. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 4851 of *Lecture Notes in Computer Science*, pages 20–27. Springer-Verlag, 2007.

[BLRF08] Daniel J. Bernstein, Tanja Lange, and Reza Rezaeian Farashahi. Binary edwards curves. In *Cryptographic Hardware and Embedded Systems – CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 244–265. Springer-Verlag, 2008.

[BN06] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-Friendly Elliptic Curves of Prime Order. In *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer-Verlag, 2006.

[Bre86] Richard P. Brent. Some integer factorization algorithms using elliptic curves. *Australian Computer Science Communications*, 8:149–163, 1986.

[BT08] Peter Birkner and Nicolas Thériault. Faster Halvings in Genus 2. To appear in *Selected Areas in Cryptography – SAC 2008*, Lecture Notes in Computer Science, Springer-Verlag, 2008.

[Can87] David G. Cantor. Computing in the Jacobian of a hyperelliptic curve. *Mathematics of Computation*, 48(177):95–101, 1987.

[CC87] David V. Chudnovsky and Gregory V. Chudnovsky. Sequences of Numbers Generated by Addition in Formal Groups and New Primality and Factorisation Tests. *Advances in Applied Mathematics*, 7:385–434, 1987.

[Cop84] Don Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Transactions on Information Theory*, 30(4):587–594, 1984.

[CY02] YoungJu Choie and Dong Kyun Yun. Isomorphism Classes of Hyperelliptic Curves of Genus 2 over $\mathbb{F}_q$. In *Information Security and Privacy – ACISP 2002*, volume 2384 of *Lecture Notes in Computer Science*, pages 190–202. Springer-Verlag, 2002.

[DI06] Christophe Doche and Laurent Imbert. Extended double-base number system with applications to elliptic curve cryptography. In *Progress in Cryptology — INDOCRYPT 2006*, volume 4329 of *Lecture Notes in Computer Science*, pages 335–348. Springer-Verlag, 2006.

[DIK06] Christophe Doche, Thomas Icart, and David R. Kohel. Ecient scalar multiplication by isogeny decompositions. In *PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 191–206. Springer-Verlag, 2006.

[DIM05] Vassil Dimitrov, Laurent Imbert, and Pradeep K. Mishra. Efficient and secure elliptic curve point multiplication using double-base chains. In *Advances in Cryptology — ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 59–78. Springer-Verlag, 2005.

[Edw07] Harold M. Edwards. A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, 44:393–422, 2007.

[FHLM04] Kenny Fong, Darrel Hankerson, Julio Lpez, and Alfred Menezes. Field Inversion and Point Halving Revisited. *IEEE Transactions on Computers*, 53(8):1047–1059, 2004.

[FR94] Gerhard Frey and Hans-Georg Rück. A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves. *Mathematics of Computation*, 62:865–874, 1994.

[Fre98] Gerhard Frey. How to disguise an elliptic curve. Talk at ECC '98, Waterloo, 1998. URL: `http://www.cacr.math.uwaterloo.ca/conferences/1998/ecc98/frey.ps`.

[Ful69] William Fulton. *Algebraic Curves*. W. A. Benjamin, New York, 1969.

[FWW06] Xinxin Fan, Thomas J. Wollinger, and Yumin Wang. Efficient Doubling on Genus 3 Curves over Binary Fields. In *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 64–81. Springer-Verlag, 2006.

[Gal01] Steven D. Galbraith. Supersingular Curves in Cryptography. In *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 495–513. Springer-Verlag, 2001.

[Gal03] Steven D. Galbraith. Weil descent of Jacobians. *Discrete Applied Mathematics*, 1:165–180, 2003.

[Gau00] Pierrick Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In *Advances in Cryptology – Eurocrypt 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 19–34. Springer-Verlag, 2000.

[Gau04] Pierrick Gaudry. Index calculus for abelian varieties and the elliptic curve discrete logarithm problem. Cryptology ePrint Archive of IACR, Report 2004/073, URL: `http://eprint.iacr.org/2004/073`, 2004. Accepted for publication in Journal of Symbolic Computation.

[GHS02] Pierrick Gaudry, Florian Hess, and Nigel P. Smart. Constructive and destructive facets of Weil descent on elliptic curves. *Journal of Cryptology*, 15(1):19–46, 2002.

[Gir08] Damien Giry. Keylength.com – Cryptographic Key Length Recommendation, September 2008. URL: `http://www.keylength.com`.

[GKP04] Cyril Guyot, Kiumars Kaveh, and Vijay M. Patankar. Explicit algorithm for the arithmetic on the hyperelliptic Jacobians of genus 3. *Journal of the Ramanujan Mathematical Society*, (19):119–159, 2004.

[GS99] Steven D. Galbraith and Nigel P. Smart. A Cryptographic Application of Weil Descent. In *Cryptography and Coding*, volume 1746 of *Lecture Notes in Computer Science*, pages 191–200. Springer-Verlag, 1999.

[GT07] Pierrik Gaudry and Emmanuel Thomé. The $\texttt{mp}\mathbb{F}_q$ library and implementing curve-based key exchanges. Proceedings of SPEED workshop, Amsterdam, 2007. URL: `http://www.hyperelliptic.org/SPEED`.

[Har97] Robin Hartshorne. *Algebraic Geometry*. Springer-Verlag, New-York, 1997.

[How93] Everett W. Howe. On the group orders of elliptic curves over finite fields. *Compositio Mathematica*, 85(2):229–247, 1993.

[JL02] Antoine Joux and Reynald Lercier. The Function Field Sieve Is Quite Special. In *Algorithmic Number Theory Symposium – ANTS V*, volume 2369 of *Lecture Notes in Computer Science*, pages 431–445. Springer-Verlag, 2002.

[JL06] Antoine Joux and Reynald Lercier. The Function Field Sieve in the Medium Prime Case. In *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 254–270. Springer-Verlag, 2006.

[JLSV06] Antoine Joux, Reynald Lercier, Nigel Smart, and Frederik Vercauteren. The Number Field Sieve in the Medium Prime Case. In *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 326–344. Springer-Verlag, 2006.

[JMS04] Michael Jacobson, Alfred J. Menezes, and Andreas Stein. Hyperelliptic curves and cryptography. In *High Primes and Misdemeanors: Lectures in Honour of the 60th Birthday of Hugh Cowie Williams*, volume 41 of *Fields Institute Communications*, pages 255–282, 2004.

[KKT05] Izuru Kitamura, Masanobu Katagi, and Tsuyoshi Takagi. A Complete Divisor Class Halving Algorithm for Hyperelliptic Curve Cryptosystems of Genus Two. In *Information Security and Privacy – ACISP 2005*, volume 3574 of *Lecture Notes in Computer Science*, pages 146–157. Springer-Verlag, 2005.

[Knu99] Erik W. Knudsen. Elliptic Scalar Multiplication Using Point Halving. In *Advances in Cryptology – ASIACRYPT99*, volume 1716 of *Lecture Notes in Computer Science*, pages 135–149. Springer-Verlag, 1999.

[Kob] Neal Koblitz. Miracles of the Height Function—A Golden Shield Protecting ECC. URL: `http://www.cacr.math.uwaterloo.ca/conferences/2000/ecc2000/koblitz.ps`. Talk at the 4th workshop on Elliptic Curve Cryptography (ECC 2000).

[Lan05] Tanja Lange. Formulae for Arithmetic on Genus 2 Hyperelliptic Curves. *Applicable Algebra in Engineering, Communication and Computing*, 15(5):295–328, 2005.

[Len87] Hendrik W. Lenstra, Jr. Factoring integers with elliptic curves. *Annals of Mathematics*, 126(3):649–673, 1987.

[LN97] Rudolf Lidl and Harald Niederreiter. *Finite Fields*, volume 20 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2nd edition, 1997.

[LS05] Tanja Lange and Marc Stevens. Efficient Doubling for Genus Two Curves over Binary Fields. In *Selected Areas in Cryptography – SAC 2004*, volume 3357 of *Lecture Notes in Computer Science*, pages 170–181. Springer-Verlag, 2005.

[McK99] James McKee. Subtleties in the Distribution of the Numbers of Points on Elliptic Curves Over a Finite Prime Field. *Journal of the London Mathematical Society*, 59(2):448–460, 1999.

[Mil86] Victor S. Miller. Short Programs for functions on Curves. IBM, Thomas J. Watson Research Center, Exploratory Computer Science, Yorktown Heights, NY, 1986. URL: `http://crypto.stanford.edu/miller/miller.pdf`.

[Mil04] Victor S. Miller. The Weil Pairing, and Its Efficient Calculation. *Journal of Cryptology*, 17(4):235–261, 2004.

[MNT01] Atsuko Miyaji, Masaki Nakabayashi, and Shunzou Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E84-A(5):1234–1243, 2001.

[Mon87] Peter L. Montgomery. Speeding the pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48(177):243–264, 1987.

[MOV93] Alfred J. Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to a finite field. *IEEE Transactions on Information Theory*, 39:1639–1646, 1993.

[Nae08] Mats Naeslund. ECRYPT Final Report on Algorithms and Key Lengths (2008). URL: `http://www.ecrypt.eu.org/documents/D.SPA.28-1.1.pdf`, August 2008. Revision 1.1.

[P1300] IEEE P1363. *Standard specifications for public key cryptography.* IEEE, 2000. URL: `http://grouper.ieee.org/groups/1363/index.html`.

[PH78] S. Pohlig and M. Hellmann. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, IT-24:106–110, 1978.

[Pol78] John M. Pollard. Monte carlo methods for index computation mod p. *Mathematics of Computation*, 32(143):918–924, 1978.

[Sch00a] Oliver Schirokauer. Using number fields to compute logarithms in finite fields. *Mathematics of Computation*, 69(231):1267–1283, 2000.

[Sch00b] Richard Schroeppel. Elliptic Curve Point Halving Wins Big. 2nd Midwest Arithmetical Geometry in Cryptography Workshop, Urbana, IL, 2000. URL: `http://www.math.uiuc.edu/~boston/magctitles.html`.

[Sha71] Daniel Shanks. Class number, a theory of factorization, and genera. In *Proceedings of Symposia in Pure Mathematics*, volume 20, pages 415–440, Providence, 1971. American Mathematical Society.

[Sha94] Igor R. Shafarevich. *Basic Algebraic Geometry 1.* Springer-Verlag, Berlin, 1994.

[Sil86] Joseph H. Silverman. *The Arithmetic of Elliptic Curves.* Springer-Verlag, New York, 1986.

[SSW96] Renate Scheidler, Andreas Stein, and Hugh C. Williams. Key-Exchange in Real Quadratic Congruence Function Fields. *Designs, Codes and Cryptography*, (7):153–174, 1996.

[Sti93] Henning Stichtenoth. *Algebraic Function Fields and Codes.* Springer-Verlag, New York, 1993.

[Suy85] Hiromi Suyama. Informal preliminary report (8). By courtesy of Richard Brent, 1985.

[SZ02] Jasper Scholten and Hui June Zhu. Hyperelliptic Curves in Characteristic 2. *International Mathematics Research Notices*, 17:905–917, 2002.

[Thé03a] Nicolas Thériault. Index Calculus Attack for Hyperelliptic Curves of Small Genus. In *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 75–92. Springer-Verlag, 2003.

[Thé03b] Nicolas Thériault. Weil Descent for Artin-Schreier Curves. Preprint, available at: `http://www.homepage.mac.com/ntheriault`, 2003.

[Was08] Lawrence C. Washington. *Elliptic Curves.* Chapman & Hall/CRC, Boca Raton, 2nd edition, 2008.

[ZD07] Paul Zimmermann and Bruce Dodson. 20 Years of ECM. In *Algorithmic Number Theory – 7th International Symposium, ANTS-VII*, volume 4076 of *Lecture Notes in Computer Science*, pages 525–542. Springer-Verlag, 2007.

A. Attacks on the Discrete-Logarithm Problem

In this appendix we discuss attacks on the discrete-logarithm problem (DLP) on general groups and on elliptic and hyperelliptic curves (i.e. on their Picard groups). At the beginning we present the algorithm of Pohlig and Hellman ([PH78]), but this not an attack but a method to split up the DLP into smaller DLPs which then can be solved using one of the other methods presented in this appendix. The next two methods are the baby-step giant-step method and Pollard's rho method which are both designed to find the discrete logarithm in general cyclic groups. After that we discuss the MOV [MOV93] and Frey-Rück [FR94] reductions as well as Weil descent (see [GS99], [GHS02] and [Gal03]). These methods can be used for elliptic curves and general abelian varieties.

A.1. Pohlig-Hellman algorithm

The Pohlig-Hellman algorithm is a method to determine the discrete logarithm in general cyclic groups. This method is mainly based on the Chinese remainder theorem. The algorithm is a method to split up the (large) discrete-logarithm problem into many smaller ones.

We start with a cyclic group G of order n with prime factorisation

$$n = p_1^{k_1} \cdot \ldots \cdot p_r^{k_r}$$

and consider the DLP

$$h = g^x \tag{A.1}$$

in the group G, where g is of order n and h an arbitrary element in G. Our goal is to determine the exponent x.

The idea of the Pohlig-Hellman strategy is to solve the DLP in all the subgroups of order $p_i^{k_i}$ of G and assemble the partial results to a solution in G. This can be done using the Chinese remainder theorem. The DLP in a subgroup of order $p_i^{k_i}$ can be, in turn, split up into even more simple DLPs in subgroups of order p_i. The DLP in the order-p_i subgroup can then be solved using other methods like Pollard rho or Shanks' baby-step giant-step algorithm.

A.1.1. Reducing the group order to prime powers

We will now describe how to reduce the first DLP into smaller DLPs in subgroups of order $p_i^{k_i}$.

To split up the main DLP into the smaller DLPs we write the exponent as $x = a_i p_i^{k_i} + b_i$, where $b_i < p_i^{k_i}$. Thus, we obtain the following equations from (A.1):

$$\begin{aligned} h^{n/p_i^{k_i}} &= (g^x)^{n/p_i^{k_i}} \\ &= g^{(a_i p_i^{k_i} + b_i) n/p_i^{k_i}} \\ &= \underbrace{g^{a_i n}}_{=1} g^{b_i n/p_i^{k_i}} \\ &= (g^{n/{p_i}^{k_i}})^{b_i}. \end{aligned} \tag{A.2}$$

Looking at the first and the last expression we get a new and particular smaller DLP, namely that of finding b_i given $h^{n/p_i^{k_i}}$ and $(g^{n/{p_i}^{k_i}})^{b_i}$ because raising h and g to the power of $n/p_i^{k_i}$ transfers the DLP into the subgroup of order $p_i^{k_i}$ where the DLP is easier to solve.

We perform the same procedure with all $p_i^{k_i}$, and we can thus formulate the following system of linear equivalences:

$$\begin{aligned} x &\equiv b_1 \pmod{p_1^{k_1}} \\ x &\equiv b_2 \pmod{p_2^{k_2}} \\ &\vdots \\ x &\equiv b_r \pmod{p_r^{k_r}}. \end{aligned}$$

Using the Chinese remainder theorem we can efficiently compute a solution x to this system, which gives us the solution of our initial DLP (A.1) in G. We remark that the Chinese remainder theorem can be applied to this situation since the moduli $p_i^{k_i}$ are pairwise coprime.

A.1.2. Reducing to prime order

It remains to explain how we can compute the exponent b_i in (A.2) explicitly. Therefore we show how to reduce a DLP in a group of order $p_i^{k_i}$ to a DLP in a subgroup of order p_i, and how a solution to this can be lifted to a solution of the DLP in the order-$p_i^{k_i}$ subgroup.

In (A.2), raising g and h to the power of $n_i/p_i^{k_i}$ ensures that both values are of order $p_i^{k_i}$ which we from now denote by $\gamma = g^{n_i/p_i^{k_i}}$ and $\psi = h^{n_i/p_i^{k_i}}$. For simplicity we also omit the fixed index i. Hence, we obtain a DLP in the subgroup of order p^k, given by the equation

$$\gamma^b = \psi,$$

where γ is of order p^k. We now write the exponent in base-p as $b = a_k p^k + a_{k-1}p^{k-1} + \ldots + a_1 p + a_0$. This gives us the following equations:

$$\begin{aligned}\psi^{p^{k-1}} = (\gamma^b)^{p^{k-1}} &= \gamma^{p^{k-1}(a_k p^k + a_{k-1}p^{k-1} + \ldots + a_1 p + a_0)} \\ &= \gamma^{p^{k-1}a_0} \underbrace{\gamma^{p^k(a_k p^{k-1} + a_{k-1}p^{k-2} + \ldots + a_1)}}_{=1}. \\ &= \gamma^{p^{k-1}a_0}\end{aligned}$$

Now, $\gamma^{p^{k-1}}$ and $\psi^{p^{k-1}}$ are of order p and computing a_0 can be done by using Pollard's rho method or Shanks' baby-step giant-step algorithm. So we assume a_0 to be known and get

$$\psi\gamma^{-a_0} = \gamma^{a_k p^k + a_{k-1}p^{k-1} + \ldots + a_1 p}.$$

Raising both sides to the power of p^{k-2} gives

$$\begin{aligned}(\psi\gamma^{-a_0})^{p^{k-2}} &= \gamma^{p^k(a_k p^{k-2} + a_{k-1}p^{k-3} + \ldots + a_2)}\gamma^{p^{k-1}a_1} \\ &= \gamma^{p^{k-1}a_1},\end{aligned}$$

since γ has order p^k, and we also get a new DLP for a_1 which can be solved the same way we solved the DLP for a_0. Repeating this for all coefficients up to a_k, the exponent $b = b_i$ of the first DLP is successfully determined.

A.2. Shanks' baby-step giant-step method

The baby-step giant-step method was introduced by D. Shanks [Sha71] to compute the class number of quadratic number fields. The method can be used to solve the discrete-logarithm problem in general groups.

One starts with a cyclic group $G = \langle g \rangle$. Let n be the order of G and $q = \lceil \sqrt{n} \rceil$. We want to find the integer x such that $g^x = h$ for a given element $h \in G$. The idea of the baby-step giant-step method is to write the exponent x as $x = dq + s$ and compute s with baby steps and d with giant steps.

We start with computing hg^{-r} for $r = 0, 1, \ldots, q-1$. If $hg^{-r} = 1$ for some r, then we are already done and $x = r$. We keep track of the values of hg^{-r} and create a list R of the pairs (hg^{-r}, r). In other words, we have

$$R = \{(hg^{-r}, r) \,|\, r = 0, 1, \ldots, q-1\}.$$

The computations of the elements of R are called *baby steps.*

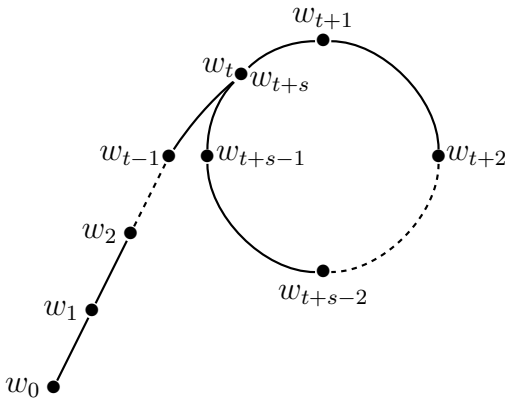
Now we compute g^{cq} for $c = 1, 2, \ldots, q$ (these calculations are called *giant steps*) and check if g^{cq} is equal to hg^{-r} for some $r \in \{0, 1, \ldots, q-1\}$. If yes, then we have $g^{cq} = hg^{-r}$ and thus $g^{cq+r} = h$. With $d = c$ and $s = r$ the discrete logarithm of h is $x = dq + s$.

This algorithm requires approximately $\sqrt{n}$ group operations and $\sqrt{n}$ storage (Section 5.2.1 in [Was08]). Therefore it is practically not feasible if the order of G is large enough.

A.3. Pollard's rho method

Here we present Pollard's rho method to compute the discrete logarithm in general cyclic groups. This method was originally designed for finding discrete logarithms in $\mathbb{F}_p^*$ [Pol78] and is based on Floyd's cycle-finding algorithm and the birthday paradox. In the following, let G be a cyclic group of order n with a generator g.

The idea is to construct a sequence $(w_i)_{i\geq 0}$ of randomly chosen group elements until a collision occurs. A collision is a pair (w_l, w_m) with $l \neq m$, such that $w_l = w_m$. The expected number of steps, that have to be performed before a collision occurs, is $\sqrt{\pi n/2}$ (cf. Section 19.5 in [ACD+05]). The sequence $(w_i)_{i\geq 0}$ is called *random walk* in G.



A pictorial description of the rho method can be given by drawing the Greek letter ρ, representing the random walk and starting at the tail with w_0. "Walking" along the line means going from w_i to w_{i+1}. If a collision occurs at w_t, then $w_t = w_{t+s}$ for some integer s, and the elements $w_t, w_{t+1}, \ldots, w_{t+s-1}$ form a loop.

To find a collision, one usually uses Floyd's cycle-finding algorithm (see Section 19.5.1.a in [ACD+05]). This idea of this algorithm is to walk along the sequence at two different speeds and hope for a collision. This can be done by using the two sequences w_i and w_{2i}. Doing a step means to increase i by 1. If $w_i = w_{2i}$ for some i, then we have found a collision.

Let h be a group element of G. We want to find an integer k such that $[k]g = h$. To solve the discrete-logarithm problem in G, i.e. computing k, we can set the random walk as

$$(w_i)_{i\geq 0} := ([a_i]g \oplus [b_i]h)_{i\geq 0}$$

for some integer sequences $(a_i)_{i\geq 0}$ and $(b_i)_{i\geq 0}$. If we have found a collision $w_l = w_m$ for $l \neq m$, then we obtain

$$[a_l]g \oplus [b_l]h = [a_m]g \oplus [b_m]h,$$

which implies $(a_l - a_m)g = (b_m - b_l)h$ and hence $k = \log_g(h) = \frac{a_l - a_m}{b_m - b_l}$.

A.3.1. Improvements and conclusion

The Pollard-Rho method can be improved by using cycle-detection tricks like Gosper's or Brent's algorithms (cf. Section 19.5 in [ACD+05]), but the time complexity does only improve by a constant factor.

Even considering all improvements to Pollard's rho method, it remains a so called "square root" algorithm, so the number of expected operations to compute the discrete logarithm remains unfeasible if the group order is large enough.

A.4. Index calculus

Index calculus is a method to compute the discrete logarithm and can be applied to some special groups. For simplicity we explain the method for the multiplicative group $\mathbb{F}_p^*$ of a finite field $\mathbb{F}_p$, where p is prime, and show how to apply this method to divisor class groups of hyperelliptic curves afterwards. This section is based on Section 5.1 in [Was08].

A.4.1. Index calculus for finite fields

Let $G = \langle g \rangle$ be a cyclic subgroup of $\mathbb{F}_p^*$ and $h \in G$ such that $h = g^k$. Let $\mathrm{DL}(h) = k$ denote the discrete logarithm of h in G. For two elements h_1 and h_2 in G we have

$$g^{\mathrm{DL}(h_1 h_2)} = h_1 h_2 = g^{\mathrm{DL}(h_1)+\mathrm{DL}(h_2)}, \tag{A.3}$$

i.e. we can write the exponents as $\mathrm{DL}(h_1 h_2) \equiv \mathrm{DL}(h_1)+\mathrm{DL}(h_2)$ modulo the group order of G. Thus, the function DL changes multiplication into addition.

The idea of index calculus is to compute $\mathrm{DL}(\ell)$ for many small prime numbers ℓ in order to compute $\mathrm{DL}(h)$ for an arbitrary value of h. We first choose a set $B = \{p_1, p_2, \ldots, p_n\}$ of small primes numbers, called factor base. Then we try to find at least $n+1$ relations of the form

$$g^{r_i} = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_n^{e_n} \quad \text{for some integers } e_1, e_2, \ldots, e_n, \tag{A.4}$$

where g^{r_i} is lifted from $\mathbb{F}_p^*$ to $\mathbb{Z}$, i.e. we consider $g^{r_i} \pmod p$ as an element of $\mathbb{Z}$. Now we solve this linear system of equations to compute the discrete logarithm (i.e. the value of DL-function) for each prime number in the factor base B. Then we can compute $h \cdot g^r$ in G for random exponents r, lift this value to $\mathbb{Z}$ and hope that it factors over the factor base. Since we know the values of the function DL for all elements in B, we can eventually compute the discrete logarithm $k = \mathrm{DL}(h)$ of h.

There are also index calculus algorithms to compute the discrete logarithm in arbitrary finite fields of the form $\mathbb{F}_{p^d}$. For small fixed values of p see e.g. [JL02], in which the running time of the algorithm is $\mathrm{L}_{p^d}[\frac{1}{3}, c]$, where $c = (32/9)^{\frac{1}{3}}$ and

$$\mathrm{L}_n[v, c] = e^{(c+o(1))(\log n)^v (\log\log n)^{1-v}}. \tag{A.5}$$

If $v = 0$, then the L-function gives a polynomial running time; for $v = 1$ it is exponential.

For the special case $\mathbb{F}_{2^d}$ we refer to [Cop84]. In this paper, Coppersmith presents an algorithm with complexity $L_{2^d}[\frac{1}{3}, c]$, where c is between $(32/9)^{\frac{1}{3}}$ and $4^{\frac{1}{3}}$. For finite fields $\mathbb{F}_q$ with $q = p^d$ and p fixed the running time of the index calculus algorithm is $L_q[\frac{1}{3}, (64/9)^{\frac{1}{3}} + o(1)]$ [Sch00a], where $o(1)$ is for $q \to \infty$. If q is a power of a medium-size prime number the same complexity (with a different constant c) was achieved using the function field sieve [JL06] and using the number field sieve [JLSV06].

A.4.2. Index calculus for hyperelliptic curves

To apply index calculus to divisor class groups of hyperelliptic curves, we have to transfer the notions of "prime" and "factor base" to divisor classes. Let $\overline{D} = [u, v]$ be a divisor class over $\mathbb{F}_q$ with $\deg(u) \leq g$, where g is the genus of the curve. We say that D is a *prime divisor* if $u(x)$ is irreducible over $\mathbb{F}_q$. To choose the factor base we select a subset of the set of prime divisors, e.g. the set of the first 50 prime divisors such that the first polynomial of each divisor in Mumford form has degree 1. Then for a divisor class $\overline{E} = [u_1, v_1]$ we test if $u_1(x)$ completely "splits" over the factor base, i.e. if $\overline{E}$ can be written as a sum of elements of the factor base. If yes, then we can analogously apply the index calculus method to find relations and compute the discrete logarithm in the divisor class group. For more details we refer to Section 13.4 in [Was08].

Genus	2	3	4	5
Square-root algorithm	q	$q^{\frac{3}{2}}$	q^2	$q^{\frac{5}{2}}$
Index calculus [Gau00]	q^2	q^2	q^2	q^2
Reduced factor base	$q^{\frac{4}{3}}$	$q^{\frac{3}{2}}$	$q^{\frac{8}{5}}$	$q^{\frac{5}{3}}$
With large primes	$q^{\frac{6}{5}}$	$q^{\frac{10}{7}}$	$q^{\frac{14}{9}}$	$q^{\frac{18}{11}}$

Table A.1.: Comparison of running times for square-root and index calculus algorithms in genus 2, 3, 4 and 5

In Table A.1 we compare the running times of square-root and index calculus algorithms for hyperelliptic curves of genus 2, 3, 4 and 5. According to the Hasse-Weil bound (see 2.15) the Picard group of a hyperelliptic curve of genus g over $\mathbb{F}_q$ has at most $(\sqrt{q}+1)^{2g}$ elements, i.e. a square-root attack in genus 2, 3, 4 and 5 requires q, $q^{\frac{3}{2}}$, q^2 and $q^{\frac{5}{2}}$ operations.

In 2000, Gaudry [Gau00] presented an index calculus algorithm, which is efficient in practise and has running time $O(q^2)$ for curves of genus less than 9 over $\mathbb{F}_q$. This method cannot outperform the square-root algorithms for genus 2 and

3, but for genus 4 the running time is equal to q^2 and even better in genus 5. The running time for the reduced factor base variant of index calculus is $O(g^5 q^{2-\frac{2}{g+1}+\varepsilon})$ and for the large-prime variant $O(g^5 q^{2-\frac{4}{2g+1}+\varepsilon})$ [Thé03a]. Looking at the two last rows of the table we see that for genus 2 the square-root algorithms perform best. For genus 3 the large-prime variant gives a very slight speedup over the square-root algorithm but this is negligible and we can still consider genus 3 safe. For the genus-4 and 5 case we see a noticeable improvement over q^2 and hence, these genera cannot be considered safe anymore.

Much research was put into finding different ways to apply index calculus methods to elliptic curves. So far none of them leads to success for curves over $\mathbb{F}_p$ or $\mathbb{F}_{2^p}$ and some evidence was given why this should be the case (cf. [Kob]).

A.5. The MOV and Frey-Rück reductions

The *MOV reduction* was proposed by Menezes, Okamoto and Vanstone [MOV93] in 1993. The idea of this technique is to transfer a discrete-logarithm problem (DLP) on an elliptic curve over $\mathbb{F}_q$ to a discrete-logarithm problem in the multiplicative group $\mathbb{F}_{q^k}^*$ using a specific isomorphism. If the value k (which is called the embedding degree of the elliptic curve) is small enough, then the discrete logarithm in $\mathbb{F}_{q^k}^*$ can be computed using index calculus methods in sub-exponential time, which is a significant improvement over for instance the square-root algorithms from the preceding sections. In this way the reduction can be seen as an attack, and therefore the name "MOV attack" is also used when talking about this method.

Luckily, it is quite easy to find elliptic curves which are not vulnerable to this kind of attack. It is very likely that a randomly chosen elliptic curve has a very large embedding degree which makes the MOV attack infeasible. Note that the transfer is always possible but the discrete-logarithm problem can only be eased via the MOV reduction if the embedding degree is sufficiently small. We will explain later what we mean by "small".

For a generalisation of the MOV reduction to hyperelliptic curves we refer to the Frey-Rück attack [FR94].

The MOV reduction is based on the use of a pairing.

Definition A.1. A *pairing* is a bilinear mapping

$$e : G_1 \times G_2 \to G,$$

where $(G_1, \oplus)$, $(G_2, \oplus)$ and $(G, \cdot)$ are groups such that

(1) $e(P \oplus Q, R) = e(P, R) \cdot e(Q, R)$ for all $P, Q \in G_1$ and $R \in G_2$;

(2) $e(P, R \oplus S) = e(P, R) \cdot e(P, S)$ for all $P \in G_1$ and $R, S \in G_2$;

(3) For all $P \in G_1$ which are different to the neutral element, there exists an element $R \in G_2$ such that $e(P, R) \neq 1$ (non-degeneracy).

A.5.1. The Weil paring

In case of the MOV reduction we utilise the Weil pairing to accomplish the transfer of the elliptic curve discrete-logarithm problem to the multiplicative group of a finite field.

Definition A.2. The *Weil pairing* is a function

$$w_n : E(\overline{\mathbb{F}}_q)[n] \times E(\overline{\mathbb{F}}_q)[n] \to \mu_n$$
$$w_n(P, Q) = \frac{f_P(\tilde{Q})}{f_Q(\tilde{P})},$$

where E is an elliptic curve defined over $\mathbb{F}_q$ and μ_n the group of nth roots of unity in $\overline{\mathbb{F}}_q^*$. The functions f_P, f_Q are elements of the function field $\overline{\mathbb{F}}_q(E)$ of E. The elements $\tilde{P}$ and $\tilde{Q}$ are divisors that are equivalent to $P - P_\infty$ and $Q - P_\infty$, respectively. Observe, that we can evaluate the functions f_P and f_Q at a divisor.

The *embedding degree* k is the smallest integer such that $\mu_n \subset \mathbb{F}_{q^k}^*$, i.e. k is minimal such that $k \mid q^k - 1$.

The Weil pairing is bilinear, non-degenerate (see [Sil86], Proposition 8.1) and computable, i.e. there exists an efficient algorithm to compute the function w_n. In particular, Miller in [Mil86] and [Mil04] shows how to compute the functions f_P and f_Q efficiently. For $P, Q \in E[n]$, the divisors $n(P - P_\infty)$ and $n(Q - P_\infty)$ are principal divisors. Hence there are functions the divisors of which are equal to $n(P - P_\infty)$ and $n(Q - P_\infty)$. These functions are precisely f_P and f_Q, i.e. $\operatorname{div}(f_P) = n(P - P_\infty)$ and $\operatorname{div}(f_Q) = n(Q - P_\infty)$. Note that we can define such a function for every n-torsion point on E.

A.5.2. The reduction

Now we show how the reduction can be used to transfer the elliptic-curve DLP to an ordinary DLP in a finite field. Let $P \in E(\mathbb{F}_q)$ be a point of order n on the elliptic curve E over $\mathbb{F}_q$ and $Q \in \langle P \rangle$. The elliptic curve discrete-logarithm problem is to find an integer λ such that $Q = [\lambda]P$.

We now compute $\alpha := w_n(P, R)$ and $\beta := w_n(Q, R)$ for an n-torsion point R with $w_n(P, R) \neq 1$ on E. Due to the bilinearity of the Weil pairing w_n we have

$$\beta = w_n([\lambda]P, R) = w_n(P, R)^\lambda,$$

which states the new discrete-logarithm problem $\alpha^\lambda = \beta$. This new problem is defined in $\mu_n \subset \mathbb{F}_{q^k}^*$.

Note that w_n implies an isomorphism

$$\langle P \rangle \to \mu_n, \quad Q \mapsto w_n(Q, R) = w_n(P, R)^\lambda$$

for a fixed n-torsion point R on E and $Q = [\lambda]P$.

To protect systems from this attack one should avoid curves with small embedding degree or at least ensure that the discrete-logarithm problem in $\mathbb{F}_{q^k}^*$ is as hard as the original problem on the elliptic curve. We remark that pairing-based cryptography uses the Weil and Tate pairings in a constructive manner. The parameters q and k are balanced so that attacks are avoided in μ_n and $E(\mathbb{F}_q)$. Constructing curves with such small values of k is a topic of active research. MNT curves [MNT01] have $k \leq 6$ while BN curves [BN06] have $k = 12$. If the pairing is not used, curves with small k should be avoided.

A.6. Weil descent

In this section we discuss Weil descent, which is a technique to transfer the discrete-logarithm problem (DLP) on an elliptic curve E over a finite field $\mathbb{F}_{q^n}$ to the Jacobian variety $\mathrm{Jac}(C)$ of a curve C of higher genus over the much smaller field $\mathbb{F}_q$. In certain situations the DLP on $\mathrm{Jac}(C)(\mathbb{F}_q)$ can be solved more easily using an index-calculus method, and this gives us the solution of the DLP on $E(\mathbb{F}_{q^n})$, which reduces the security of the elliptic curve DLP.

Using Weil descent to transfer the elliptic curve DLP to another abelian variety, e.g. to the Jacobian of a higher genus curve was first proposed by Frey [Fre98]. The method was further developed by Galbraith and Smart [GS99] and by Gaudry, Hess and Smart [GHS02]. In [Gal03] Galbraith has shown that the technique of Weil descent can also be applied to the case of the discrete-logarithm problem in the Jacobian of curves of genus greater than one.

Weil restriction

We will now give the idea of how to use Weil descent to transfer the DLP of an elliptic curve over a field K to an abelian variety defined over a subfield of K.

Let $k = \mathbb{F}_q$ be a finite field of odd characteristic p and $q = p^r$ for $r > 1$. Let $K = \mathbb{F}_{q^n}$ be a field extension of degree n over k. Now we consider an elliptic curve over K given by the equation

$$E : Y^2 = X^3 + \alpha X + \beta, \tag{A.6}$$

where $\alpha, \beta \in K$. If the characteristic of k is 2, then we need to use a more general form of the curve equation.

The field K is a k-vector space of dimension n. So let $\{\theta_0, \theta_1, \ldots, \theta_{n-1}\}$ be a k-basis of K. We write

$$X = \sum_{i=0}^{n-1} x_i\theta_i, \quad Y = \sum_{i=0}^{n-1} y_i\theta_i, \quad \alpha = \sum_{i=0}^{n-1} a_i\theta_i \quad \text{and} \quad \beta = \sum_{i=0}^{n-1} b_i\theta_i,$$

where $a_i, b_i \in k$.

Now, we substitute these expressions into the curve equation (A.6) and equate the coefficients of $\theta_0, \theta_1, \ldots, \theta_{n-1}$. We obtain n equations in the $2n$ unknowns $\{x_0, x_1, \ldots, x_{n-1}, y_0, y_1, \ldots, y_{n-1}\}$. By means of these relations we are given an abelian variety A over the smaller field k of dimension n. This variety is called *Weil restriction of scalars of* E, and the procedure of creating this variety from the curve E is called *Weil descent.*

The variety A over k is isomorphic (as a group) to the elliptic curve E over K, and we can perform the group law on $A(k)$ by converting a point

$$(x_0, x_1, \ldots, x_{n-1}, y_0, y_1, \ldots, y_{n-1}) \in A(k)$$

to the point

$$(x_0\theta_0 + x_1\theta_1 + \ldots + x_{n-1}\theta_{n-1}, y_0\theta_0 + y_1\theta_1 + \ldots + y_{n-1}\theta_{n-1}) \in E(K).$$

Now, we can use the addition formulas on E.

Let us assume the elliptic curve $E(K)$ contains a subgroup of large prime order ℓ. Since $A(k)$ is isomorphic to $E(K)$, the abelian variety A contains an irreducible subvariety B which, in turn, contains a subgroup of order ℓ. To solve the DLP in the order ℓ subgroup of $E(K)$ we can now try to solve the DLP in the subvariety B, or more precisely in its order ℓ subgroup. To do so we find a curve C on the variety A the Jacobian of which contains a subvariety which is isogeneous to B. The dimension of the abelian variety A is n. To obtain a curve (which is a variety of dimension 1) we have to reduce the dimension of A. This can be done for example by intersecting A with all hyperplanes which pass through the zero element of A. Therefore we set $x_0 = x_1 = \ldots = x_{n-1}$. If we apply this to the set of relations generating A, we reduce the dimension to 1 and get a curve with the desired properties. Since $\langle P \rangle$ has to be contained in $\mathrm{Div}_k^0(C)$ the genus of C is at least n; for most curves it is much larger. If the genus of this curve is not too large, it can be possible to solve the DLP using index calculus methods in $\mathrm{Div}_k^0(C)$. The transfer from the elliptic curve can be accomplished via the *conorm-norm homomorphism* $\phi : E(K) \to \mathrm{Div}_k^0(C)$ (see Section 3.6 in [GHS02] for details).

Remarks and conclusion

We have to point out that an attacker is completely free in choosing a curve within the variety $A(k)$. Intersecting the variety with hyperplanes is just one way to reduce its dimension and to get a curve with the desired properties.

Since the abelian variety A is defined over a subfield of K, it is obvious that this method cannot be applied when the curve E is defined over $\mathbb{F}_p$, where p is prime. For fields $\mathbb{F}_{2^p}$, where p is prime, there exists no intermediate field, so the Weil descent has to go to $\mathbb{F}_2$. In general, the genus of a curve on $A(k)$ grows exponentially with the extension degree. However, it could be that for some curves over $\mathbb{F}_{2^p}$ there exists a Weil descent process, that is successful and leads to a curve of genus p over $\mathbb{F}_2$. So far the only susceptible primes are Mersenne primes, e.g. $p = 127$.

Acknowledgements

First of all, I would like to express my gratitude to my supervisor Tanja Lange. She invested much time and effort into the supervision of my work, and provided interesting research problems, which have led to numerous results and many publications. Her knowledge and her directions had significant influence upon my work and the advancement of my education.

I would also like to thank Daniel J. Bernstein for the always valuable discussions, comments and hints.

For the very interesting and fruitful collaboration that has led to two scientific papers and to the results in Chapters 3 and 4, I would like to thank Nicolas Thériault very much.

I am grateful to my reading committee, consisting of Tanja Lange, Daniel J. Bernstein, Gerhard Frey, Steven Galbraith and Henk van Tilborg. Their comments and suggestions helped me to improve the thesis significantly.

I would also like to thank my colleagues Michael Naehrig, Christiane Peters and Peter Schwabe who have proofread the manuscript of my thesis and gave valuable comments.

I thank Pierrick Gaudry for the explanations of the Weil descent and for proofreading Section A.6 of the thesis. I also thank Elisa Gorla for the discussion about blow-up techniques.

Summary: Efficient Arithmetic on Low-Genus Curves

Public key cryptosystems are almost always based on two problems in number theory, the discrete-logarithm problem and the factorisation of integers. In this thesis we treat certain aspects of both of these problems.

The most crucial parts of a cryptosystem that is based on the discrete-logarithm problem are the group and the efficiency of the arithmetic in this group. In this work we have investigated divisor class groups of hyperelliptic curves of genus 2 and 3 over binary fields. We suggest certain curves such that the appropriate group is considered secure, and provide efficient arithmetic on these curves.

The most important operation in curve-based cryptosystems is single-scalar multiplication of divisor classes. Therefore a very time-efficient arithmetic is necessary. Since scalar multiplication is almost always computed using double-and-add algorithms (or variants of these), it stands to reason to develop efficient doubling and addition formulas. In case of elliptic curves it turned out that point halving is very efficient, and hence halve-and-add algorithms proved very successful and could even replace the double-and-add methods in some situations. So it is natural to ask if similar results can be obtained for hyperelliptic curves as well. For genus-2 curves we have developed explicit halving formulas which can in some settings even beat the doubling counterparts. For the high-speed case on the genus-2 curves we also give a complete case study, that covers all special cases, depending on the polynomial representation of the divisor class.

We have generalised this also to the genus-3 case and investigated several types of curves and developed explicit halving formulas. For some curves of a rather general form we could even beat the doubling formulas by 10 to 20 field multiplications which is a speedup of about 30-40%. For the most common setting in genus 3 we give (like in genus 2) a complete case study for all possible subcases. This provides a programmer with everything he needs to do an implementation of a cryptosystem based on the DLP on divisor class groups of hyperelliptic curves of genus 3.

The third subject in this thesis (besides hyperelliptic curves of genus 2 and genus 3) is Edwards curves. We have investigated elliptic curves in Edwards and twisted Edwards form. We have looked at explicit addition, doubling and tripling formulas in affine, projective and inverted Edwards coordinates. The arithmetic on Edwards curves turns out to be faster than on elliptic curves in other forms. Twisted Edwards curves cover even more elliptic curves: We have shown that every Montgomery-form elliptic curve is birationally equivalent to a

twisted Edwards curve. This brings the speed of the Edwards addition law to Montgomery curves.

Furthermore, we have demonstrated how to construct Edwards and twisted Edwards curves with prescribed torsion subgroup and positive rank, which is essential for the ECM method of factorisation. With this we treat the second problem on which cryptosystem can be based on. The use of Edwards curves improved the speed of factoring integers by using better curves and faster arithmetic.

Curriculum Vitae

Peter Birkner was born on 26 March 1976 in Hamminkeln, Germany.

In 1998, he started his studies in mathematics and computer science at University of Paderborn, Germany. He graduated in 2004. His master's thesis was entitled "On the Determination of Galois Groups". During his studies, he worked as a teaching assistant, leading problem sessions for several mathematics courses at the university.

He then started his PhD studies at RWTH Aachen University, Germany. In 2006, he transferred to Technical University of Denmark, where he worked under the supervision of Prof. Dr. Tanja Lange. In the same year he was invited for a research stay at the Fields Institute in Toronto, Canada and visited there for three months. In 2007, he followed his supervisor to Technische Universiteit Eindhoven, The Netherlands. From 2007 to 2009, he was PhD student in the coding theory and cryptology group at the same university, under the supervision of Prof. Dr. Tanja Lange. The present thesis contains most of the results of his work from 2004 to 2009.

His research interests are in the area of algebraic number theory and cryptography, in particular efficient arithmetic in divisor class groups of hyperelliptic curves over binary fields, Edwards curves and integer factorisation methods.

Index