Efficient Divisor Class Halving on Genus Two Curves

Peter Birkner [*]

Department of Mathematics, Technical University of Denmark (DTU)
Matematiktorvet, Building 303, DK-2800 Kongens Lyngby, Denmark
peter@mat.dtu.dk

Abstract. Efficient halving of divisor classes offers the possibility to improve scalar multiplication on hyperelliptic curves and is also a step towards giving hyperelliptic curve cryptosystems all the features that elliptic curve systems have. We present a halving algorithm for divisor classes of genus 2 curves over finite fields of characteristic 2. We derive explicit halving formulae from a doubling algorithm by reversing this process. A family of binary curves, that are not known to be weak, is covered by the proposed algorithm. Compared to previous known halving algorithms, we achieve a noticeable speed-up for this family of curves.

Keywords. hyperelliptic curve, divisor class halving, binary fields.

1 Introduction

Since hyperelliptic curve cryptosystems (HECC) gain similar attention as their elliptic counterparts, it is very interesting to investigate, whether ideas and methods can be transferred from the elliptic to the hyperelliptic case. The most important operation used by elliptic curve cryptosystems (ECC) is scalar multiplication which is composed of point addition and doubling (when using an double-and-add algorithm) or point addition and halving (when using an half-and-add algorithm [7,11]). These operations are well investigated and it is likely that the present formulae are the most efficient ones. For HECC explicit formulae for addition, doubling and hence scalar multiplication of divisor classes are also known [1,8].

Efficient halving of divisor classes offers the possibility to improve scalar multiplication on hyperelliptic curves and is also a step towards giving HECC all the features that ECC have. Halving a divisor class of a hyperelliptic curve is the reverse operation to doubling, i. e. given a divisor class $\overline{D}$ one computes another divisor class $\overline{E}$ such that $2\overline{E} = \overline{D}$ or written in slightly informal notation: $\frac{1}{2}\overline{D} = \overline{E}$.

In this paper we present an efficient divisor class halving algorithm for hyperelliptic curves of genus 2 over finite fields of characteristic 2. Covering a large family of curves, that are of cryptographic interest, the complexity of our algorithm (1I, 8M, 5SR, 2S, 1HT, 1TR)[1] is only less higher than the complexity of the fastest doubling algorithm (1I, 5M, 6S) for divisor classes [9]. The proposed halving method is based on explicit doubling formulae [9] that we use to develop the halving formulae.

The first divisor class halving algorithm for binary curves was proposed by Kitamura, Katagi and Takagi [5,6]. To our knowledge this is the only result on halving of divisor classes so far. Their method covers the general case as well as some exceptional cases which do occur with very low

[*] Supported by the Danish Research Council for Technology and Production Sciences, grant no. 274-05-0151.

[1] To describe the complexity of an algorithm we use the following abbreviations: I – Inversion, M – Multiplication, S – Squaring, SR – Square Root, TR – Trace, HT – Half Trace.

probability. In the general case their complexity is 1I, 18M, 2SR, 2S, 2HT, 2TR in the best case and 1I, 21M, 3SR, 2S, 2HT, 2TR in the worst case.

The special class of curves considered in Appendix D of [6] is covered by our study. However, instead of having a one-parameter family our curves has two free parameters. In the worst case our complexity is 1I, 8M, 5SR, 2S, 1HT, 1TR which is significantly faster even than their formulae.

The remainder of this paper is structured as follows: In the next section we recall some important notions and make a classification of genus 2 curves. In Section 3 we show how we developed the halving formulae by reversing the doubling formulae. The following section contains the actual halving algorithm.

2 Basic Notations and Preliminaries

In this section we briefly recall the definitions of hyperelliptic curves, divisor class groups and the Mumford representation. We also make a classification of hyperelliptic curves over finite fields of characteristic 2 because we focus on a family of curves in this paper. Within the halving algorithm we need to solve a quadratic equation in a finite field of even characteristic. So we will explain how to compute solutions for this at the end of this section.

A comprehensive source for the mathematics of finite fields is [10]. For background on hyperelliptic curves we refer the interested reader to [1], from which the following definitions and notations are taken.

Definition 1 (Hyperelliptic curve). *Let K be a field and let $\overline{K}$ be the algebraic closure of K. A curve C, given by an equation of the form*

$$C : y^2 + h(x)y = f(x), \tag{1}$$

where $h \in K[x]$ is a polynomial of degree at most g and $f \in K[x]$ is a monic polynomial of degree $2g+1$, is called a hyperelliptic curve of genus g over K *if no point on the curve over $\overline{K}$ satisfies both partial derivatives $2y + h = 0$ and $f' - h'y = 0$.*

The last condition ensures that the curve is nonsingular. In this paper we concentrate on hyperelliptic curves of genus 2 over finite fields of characteristic 2. In this case we need a non-zero polynomial h in the curve equation as will be shown now according to [1, p. 309].

Assuming $h = 0$, the partial derivative for y is equal to zero and the one for x is equal to $f'(x) = 0$, which has $2g$ roots in $\overline{K}$. Let x_1 be one of them. Then we can find an element $y_1 \in \overline{K}$ such that $f(x_1) = y_1^2$ which leads to a singular point $P = (x_1, y_1)$ satisfying the curve equation and both partial derivatives. Hence, there is no hyperelliptic curve with $h = 0$ over a field of characteristic 2.

Definition 2 (Divisor class group). *Let C be a hyperelliptic curve of genus g over a field K. The group of degree zero divisors of C is denoted by* Div_C^0. *The quotient group of* Div_C^0 *by the group of principal divisors of C is called the* divisor class group of C *and is denoted by* Pic_C^0. *It is also called the* Picard group of C.

Theorem 1 (Mumford). *Let C be a hyperelliptic curve of genus g over an arbitrary field K. Each nontrivial divisor class of C over K can be represented by a unique pair of polynomials $u, v \in K[x]$, where*

1. *u is monic,*
2. $\deg v < \deg u \leq g$,

3. $u \mid v^2 + vh - f$.

Our proposed halving algorithm in Section 4 expects the input divisor class to be in Mumford representation and works directly on the coefficients of the polynomials u and v. The resulting divisor class is also given in the Mumford form.

2.1 Classification of Genus Two Curves

In this paper we deal with hyperelliptic curves of genus 2 over $\mathbb{F}_{2^d}$. To avoid Weil descent attacks [4] one usually restricts to prime degree field extensions for cryptographic applications. So in the following we particularly assume d to be odd.

The genus 2 curves over $\mathbb{F}_{2^d}$ can be sorted into three different categories depending on the 2-rank of the divisor class group of the curve (see [3]). All points P in the support of a divisor class of order 2 satisfy $P = \iota(P)$, where ι is the hyperelliptic involution. If $P = (x, y)$ is not the point at infinity, its coordinates must satisfy $y = h(x) + y$. So, x is a root of $h(x)$ and we have that the degree of h equals the 2-rank of Pic^0_C. In this paper we focus on curves whose divisor class group has 2-rank equal to one, i. e. in the curve equation we have $\deg h = 1$. In [2,1] these curves are called curves of Type II. Over a field $\mathbb{F}_{2^d}$ with d odd, one can perform the following transformations

$$x \mapsto \mu^2 x' + \lambda \quad \text{and} \quad y \mapsto \mu^5 y' + \mu^4 \alpha x'^2 + \mu^2 \beta x' + \gamma,$$

where μ is such that $\mu^3 = h_1$, $\lambda = h_0 h_1^{-1}$, $\alpha = \sqrt{\lambda + f_4}$, β a root of $x^2 + h_1 x + f_2 + f_3\lambda + \varepsilon h_1^2$ with $\varepsilon = \mathrm{Tr}\left((f_2 + f_3\lambda)h_1^{-2}\right)$ and $\gamma = (\lambda^2 f_3 + \lambda^4 + f_1)h_1^{-1}$, to obtain a unique representative of each isomorphism class given by

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

where $f_2 \in \mathbb{F}_2$ and $f_0, f_3 \in \mathbb{F}_{2^d}$ [1, Proposition 14.37]. So, for the remainder of this paper we consider (2) as a Type II curve.

Since $f_2 \in \mathbb{F}_2$, we have $2 \cdot 2^d \cdot 2^d = 2^{2d+1}$ different choices for the right-hand side of (2), i. e. Type II covers (up to isomorphism) 2^{2d+1} different curves of genus 2 where our halving algorithm can be applied.

2.2 Quadratic Equations in $\mathbb{F}_{2^d}$

In the halving algorithm we need to solve a quadratic equation in $\mathbb{F}_{2^d}$. Provided a solution exists, we can use a simple formula to compute it.

Consider the quadratic equation $X^2 + aX + b = 0$ over the finite field $\mathbb{F}_{2^d}$. By substituting X by X/a, one gets the simpler equation

$$T^2 + T = c, \quad \text{with } c = b/a^2, \tag{3}$$

which has a solution in the field $\mathbb{F}_{2^d}$ if and only if the trace of c is equal to zero [1, Section 11.2.6]. If d is odd, then a solution of (3) is given by

$$t = \sum_{i=0}^{(d-3)/2} c^{2^{2i+1}}. \tag{4}$$

When t is one solution of the quadratic equation (3), then $t + 1$ is the other one. See [1, Section 11.2.6] for details.

2.3 Choice of the Field Representation

Like in the doubling formulae we have to compute inversions, multiplications and squarings to halve a divisor class. Additionally we need to be able to efficiently compute square roots, traces and half-traces. In order to speed these operations up, one can use a normal basis representation. Having this we can compute the square of a field element simply by shifting the representing vector. Computing a square root works the same way but shifting to the opposite direction. Because traces and half-traces are sums of powers of squares, they can be calculated very efficiently, too. In a hardware implementation, multiplications and inversions in the field can be hard-coded in order to get best performance.

Software libraries like NTL work with a polynomial basis representation and do not provide efficient square root computations in characteristic two. So, we implemented our own square root function for the finite field $\mathbb{F}_{2^{83}}$ and present some timings for field operations. We used the Number Theory Library (NTL 5.4) together with the GNU Multiple Precision Arithmetic Library (GMP 4.2.1) and the GNU Compiler Collection (GCC 4.0.1) on an Apple MacBook with a 2,0 GHz Intel Dual Core CPU to compute the benchmarks. The multiplication, inversion and squaring functions are taken from NTL, the square root function is our own implementation for that particular finite field. We measured the time for 100,000 operations each.

Operation	Time [sec.]	# of Multiplications
M	0.065966	1.00
I	0.52136	7.90
SR	0.3775	5.72
S	0.045714	0.69

Table 1. Timings of field operations in $\mathbb{F}_{2^{83}}$

3 From Doubling to Halving

In this section we derive the halving formulae from the doubling formulae. Therefore, we present first how to double a divisor class given in Mumford representation using explicit formulae. Then we explain how we found the halving formulae by reversing the doubling algorithm.

In the entire section we assume C to be a Type II hyperelliptic curve of genus 2 over $\mathbb{F}_{2^d}$, where d is odd, given by equation (2). In the following we also need to assume that the group order of $\mathrm{Pic}^0_C(\mathbb{F}_{2^d})$ is $2r$, where r is odd.[2] For cryptographic applications one wants to work in a cyclic subgroup of prime order l. So we denote the order l-subgroup of $\mathrm{Pic}^0_C(\mathbb{F}_{2^d})$ by S. Note, however, that the following considerations also hold muta mutandis in the subgroup of order r.

3.1 Doubling of Divisor Classes

Let $\overline{E} = [x^2 + u'_1x + u'_0,\, v'_1x + v'_0]$ be a divisor class in the order l-subgroup S of $\mathrm{Pic}^0_C(\mathbb{F}_{2^d})$. Because l is prime, there exists no proper subgroup of S and hence it is cyclic. So, each divisor class contained in S is the double of another divisor class in the same subgroup, i. e. each divisor class in S can be

[2] This ensures that there is no element of order 4.

doubled and hence also be halved. In [5] the elements of this subgroup are called *proper divisor classes.*

We can compute the doubled divisor class $\overline{D} = 2\overline{E} = [x^2 + u_1x + u_0,\, v_1x + v_0]$ using Lange and Steven's explicit formulae (see [9]):

$$u_1 = \left(\frac{{u'_0}^2}{f_0 + {v'_0}^2}\right)^2, \tag{5}$$

$$u_0 = \left(\left({u'_1}^2 + f_3\right)\left(\frac{{u'_0}^2}{f_0 + {v'_0}^2}\right) + u'_1\right)^2 + \left(\frac{{u'_0}^2}{f_0 + {v'_0}^2}\right), \tag{6}$$

$$v_0 = \left(\frac{{u'_0}^2}{f_0 + {v'_0}^2} + {u'_1}^2 + f_3\right)u_0 + {u'_0}^2, \tag{7}$$

$$\begin{aligned} v_1 = &\left(\frac{{u'_0}^2}{f_0 + {v'_0}^2} + {u'_1}^2 + f_3\right)\left({u'_1}^2 + f_3\right)\left(\frac{{u'_0}^2}{f_0 + {v'_0}^2}\right) \\ &+ \left(\frac{{u'_0}^2}{f_0 + {v'_0}^2}\right)u_1 + f_2 + {v'_1}^2. \end{aligned} \tag{8}$$

Notice that we are considering a curve of form (2), i. e. $h(x) = x$. Hence, the coefficient h_1 occurring in Lange and Steven's formulae equals one and does not appear here.

3.2 Halving of Divisor Classes

Now, we turn around and compute the half of a divisor class by applying the doubling formulae in reverse order. Given a divisor class $\overline{D} = [x^2 + u_1x + u_0,\, v_1x + v_0]$, we show how to compute a divisor class $\overline{E} = [x^2 + u'_1x + u'_0,\, v'_1x + v'_0]$ such that $\overline{D} = 2\overline{E}$. Therefore, we need again to say that $\overline{D}$ must be contained in the order l-subgroup S of $\mathrm{Pic}^0_C(\mathbb{F}_{2^d})$ in order to ensure that the double and the half of each divisor class does exist in this particular subgroup.

Taking $\sqrt{u_1}$ from (5), we can write u_0 using (6) as

$$u_0 = \left(({u'_1}^2 + f_3)\sqrt{u_1} + u'_1\right)^2 + \sqrt{u_1}. \tag{9}$$

Now, using the fact that we are in characteristic 2 we can expand the quadratic expression and arrange the terms such that we get a quartic equation in u'_1 on the right-hand side:

$$u_0 = {u'_1}^4 u_1 + {u'_1}^2 + \left(f_3^2 u_1 + \sqrt{u_1}\right). \tag{10}$$

Substituting $U = {u'_1}^2$ yields a quadratic equation in the variable U:

$$U^2 u_1 + U = u_0 + f_3^2 u_1 + \sqrt{u_1}. \tag{11}$$

Multiplying both sides by u_1 and substituting $T = U \cdot u_1$ afterwards yields a quadratic equation $T^2 + T = c$ where $c = u_1u_0 + f_3^2u_1^2 + u_1\sqrt{u_1}$ like (3).

Because $\overline{D}$ is an element of the subgroup S, there exists an element $\overline{E} \in S$ with $\overline{D} = 2\overline{E}$. So u_1, u_0, v_1 and v_0 can be written as in (5), (6), (7) and (8). Hence, we know that there exists a solution of $T^2 + T = c$ (because this equation holds if and only if (6) holds). Due to Section 2.2 this implies that the trace of c is equal to zero. We also know that there exist two solutions t and $t+1$ which can be computed using (4). After adjusting these two solutions by dividing by u_1 we have two solutions of (11). Re-substituting $u'_1 = \sqrt{t/u_1}$ or $u'_1 = \sqrt{(t+1)/u_1}$ respectively yields two possible values for u'_1 in (10). We will show how to figure out which of these two solutions is the right one at the end of this section. For now let us suppose that we already know the correct u'_1.

Taking v_0 from (7) and writing again $\frac{{u'_0}^2}{f_0+{v'_0}^2}$ as $\sqrt{u_1}$, we obtain:

$$v_0 = \left(\sqrt{u_1} + {u'_1}^2 + f_3\right) u_0 + {u'_0}^2, \tag{12}$$

which leads us to a new expression for u'_0 using the already known value u'_1:

$$u'_0 = \sqrt{v_0 + \left(\sqrt{u_1} + {u'_1}^2 + f_3\right) u_0}. \tag{13}$$

Now we are able to compute v'_0 using u'_0 and (5):

$$v'_0 = \sqrt{\frac{{u'_0}^2}{\sqrt{u_1}} + f_0}. \tag{14}$$

The last step is to compute v'_1 using (8):

$$v'_1 = \sqrt{v_1 + \sqrt{u_1}\left((\sqrt{u_1} + {u'_1}^2 + f_3)({u'_1}^2 + f_3) + u_1\right) + f_2}. \tag{15}$$

Let us now come back to figuring out which of the two solutions of the quadratic equation $T^2 + T = c$ is the right one. In order to do that, we use the first solution t and continue computing the halved divisor class as explained above. If this choice was correct, then the halved divisor class is a proper one, i. e. it is contained in the subgroup S. So we have to check this. As we have seen above, the trace of $c = u_1^2\left(\frac{u_0}{u_1} + f_3^2 + \frac{\sqrt{u_1}}{u_1}\right) = u_1\left(u_0 + u_1 f_3^2 + \sqrt{u_1}\right)$ is zero if and only if the divisor class is contained in S. We now check if the obtained divisor class can be halved, i. e. whether $\mathrm{Tr}\left(u'_1(u'_0 + u'_1 f_3^2 + \sqrt{u'_1})\right) = 0$. If this holds, then the first solution t was correct and we have computed the correct halved divisor class. If the trace is not zero, we use the other solution $t+1$ of the quadratic equation. So the trace serves as a criteria to determine the right solution of (11). Note, that this test involves computing u'_1 and u'_0. So it should be performed as soon as they are computed. If the other solution turns out to be the correct one, we have to redo the computation of u'_1 and u'_0 using $t+1$ instead of t.

After computing u'_1, u'_0, v'_1 and v'_0 we can write the halved divisor class in Mumford representation: $E = [x^2 + u'_1 x + u'_0,\ v'_1 x + v'_0]$.

3.3 The Case $u_1 = 0$

The formulae presented in the previous section hold in the generic case, i. e. if both the input and output have u and v of full degree and no zero coefficients. To complete the above study we now consider how to compute the half of a divisor class with $u_1 = 0$.

The other cases appear with very low probability and do not belong to the main algorithm. Implementers going for an implementation of all possible cases should consult [9] and [8] for a complete case study.

We now consider the divisor class $\overline{D} = [x^2 + u_0,\, v_1x + v_0]$, where u_1 equals zero. From equation (5) follows directly $u'_0 = 0$. Using this and equation (6) we get $u_0 = u'^2_1$ and hence, $u'_1 = \sqrt{u_0}$. This shrinks (8) to $v_1 = f_2 + v'^2_1$ and we have $v'_1 = \sqrt{v_1 + f_2}$. Having $u'_0 = 0$, one can see that the four equations (5), (6), (7) and (8) do not depend on v'_0 any longer, so this value becomes arbitrary. So, (14) should not be performed.

The complete procedure explained above leads to the actual halving algorithm presented in the next section.

4 The Divisor Class Halving Algorithm

We present an efficient divisor class halving algorithm for genus 2 curves of Type II (cf. Section 2.1) over $\mathbb{F}_{2^d}$, where d is odd, based on the formulae derived in the previous section. We do not follow the steps literally but change them to allow more efficient computations.

We shortly repeat the prerequisites: The curve parameters must be chosen such that the order of $\mathrm{Pic}^0_C(\mathbb{F}_{2^d})$ is equal to $2r$ for an odd number r. The input divisor class must be contained in the order l-subgroup of $\mathrm{Pic}^0_C(\mathbb{F}_{2^d})$, where l is prime.

Algorithm 1 Divisor Class Halving (HLV)

```
INPUT:  Divisor class $\overline{D} = [u, v]$, where $u = x^2 + u_1x + u_0,\ v = v_1x + v_0$ and the
        pre-computed values $f_3^2, \sqrt{f_0}$
OUTPUT: Halved divisor class $\overline{E} = [u', v']$ such that $\overline{D} = 2\overline{E}$
 1: $q_1 \leftarrow \sqrt{u_1},\ q_2 \leftarrow 1/q_1,\ q_3 \leftarrow q_2^2,\ q_4 \leftarrow u_0q_3,\ q_5 \leftarrow \sqrt{q_2}$        ▷ 1I, 1M, 2SR, 1S
 2: $q_6 \leftarrow \sqrt{q_4},\ c \leftarrow u_1(q_6 + q_5 + f_3)$                                                           ▷ 1SR, 1M
 3: $t' \leftarrow \sum_{i=0}^{(d-3)/2} c^{2^{(2i+1)}}$                                                                          ▷ 1HT
 4: $u'_1 \leftarrow t'q_2,\ t \leftarrow u'^2_1,\ s_1 \leftarrow v_0 + (q_1 + t + f_3)u_0$                                     ▷ 2M, 1S
 5: $u'_0 \leftarrow \sqrt{s_1},\ b \leftarrow \mathrm{Trace}(u'_1(u'_0 + t + f_3))$                                         ▷ 1M, 1SR, 1TR
 6: if $b = 0$ then
 7:     $v'_0 \leftarrow q_5u'_0 + \sqrt{f_0}$                                                                                ▷ 1M
 8: else
 9:     $t \leftarrow t + q_3,\ u'_1 \leftarrow u'_1 + q_2$
10:     $u'_0 \leftarrow u'_0 + q_6,\ v'_0 \leftarrow q_5u'_0 + \sqrt{f_0}$                                                  ▷ 1M
11: end if
12: $v'_1 \leftarrow \sqrt{v_1 + q_1\Big((q_1 + t + f_3)(t + f_3) + u_1\Big) + f_2}$                                          ▷ 2M, 1SR
13: return $[x^2 + u'_1x + u'_0,\ v'_1x + v'_0]$                                                         ▷ Total: 1I, 8M, 5SR, 2S, 1HT, 1TR
```

We now explain the steps of the algorithm according to the formulae derived in the previous section.

Some expressions in the halving formulae do occur more than once. To avoid recomputations, we replace them by $q_1, \ldots, q_6$ in Step 1 and 2. To solve the quadratic equation (11) we have to compute $u_1(u_0 + u_1f_3^2 + \sqrt{u_1})$. What we actually do in the algorithm is computing $u_1(q_6 + f_3 + q_5)$ which is the square root of $u_1(u_0 + u_1f_3^2 + \sqrt{u_1})$ and use that $\mathrm{Tr}(a^2) = \mathrm{Tr}(a)$ for any $a \in \mathbb{F}_{2^d}$. The reason for this is that we can save 1SR since u'_1 is root of the (via multiplication by q_2) adjusted solution t'. In Step 3 the solution of the quadratic equation is computed as in (4) and then adjusted in Step 4. In Step 5 the value u'_0 is computed according to (13).

According to the explanation at the end of the previous section, we now have to compute the trace of $u_1'(u_0' + \sqrt{u_1'} + u_1' f_3^2)$ in order to perform the check whether we calculated the right solution of the quadratic equation or not. To reduce the number of operations we compute the trace of $u_1'(u_0' + t + f_3)$ instead. Doing this saves 1M, 1SR and 1S. To see that these two traces are equal, we point out that $\mathrm{Tr}(u_1'\sqrt{u_1'}) = \mathrm{Tr}(u_1' t)$, $\mathrm{Tr}({u_1'}^2 f_3^2) = \mathrm{Tr}(u_1' f_3)$ and that the trace is a linear map.

In Steps 6 to 11 we compute v_0' depending on the trace b. If this trace is equal to zero, we continue by computing v_0' using (14). For $b = 1$ we use $t' + 1$ instead of t' in Step 3. Hence, we have to adjust x by adding q_3, u_1' by adding q_2 and u_0' by adding q_6 in Steps 9 and 10. After that we can compute v_0' in the same way as in Step 7.

The last thing to do is computing v_1' using (15) in Step 12. Finally the algorithm returns the desired halved divisor class in Mumford representation. The steps, considered so far, have a total complexity of 1I, 8M, 5SR, 2S, 1HT, 1TR in both cases $b = 1$ and $b = 0$.

5 Conclusion and Outlook

In this paper we presented an efficient halving algorithm for divisor classes of a family of hyperelliptic curves of genus 2 over binary fields. Compared to the previous result by Kitamura, Katagi and Takagi [5,6] we gained a notable speed-up for this family (see Table 2).

In Appendix D of [6] the authors consider curves of form

$$y^2 + xy = x^5 + f_1 x + f_0.$$

The transformation $y \mapsto \tilde{y} + f_1$ maps to the isomorphic curve $\tilde{y}^2 + x\tilde{y} = x^5 + \tilde{f}_0$, which is a Type II curve. This equation shows that their family of curves has only one parameter that can be chosen freely while our family achieves full generality for Type II curves needing far less operations (cf. Table 2).

	Type II curve (see Section 2.1) $y^2 + xy = x^5 + f_3x^3 + f_2x^2 + f_0$	**Special curve** (see [6], Appendix D) $y^2 + xy = x^5 + f_0$
Kitamura, Katagi, Takagi [5,6]	1I, 15M, 3SR, 3S, 2HT, 2TR	1I, 12M, 5SR, 2S, 1HT, 1TR
This work	1I, 8M, 5SR, 2S, 1HT, 1TR	1I, 8M, 5SR, 2S, 1HT, 1TR

Table 2. Complexity of halving algorithms in worst case

We would like to point out that we can improve the efficiency of our algorithm as well as that of doubling by leaving out the computation of v_1 in a Montgomery like scalar multiplication, since the new values u_0, u_1 and v_0 do not depend on it. We are investigating the required addition formulae.

References

1. Roberto Avanzi, Henri Cohen, Christophe Doche, Gerhard Frey, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *The Handbook of Elliptic and Hyperelliptic Curve Cryptography*. CRC Press, 2005.
2. Bertrand Byramjee and Sylvain Duqesne. Classification of genus 2 curves over $\mathbb{F}_{2^n}$ and optimization of their arithmetic. Cryptology ePrint Archive of IACR, Report 2004/107, 2004.

3. 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.
4. 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.
5. 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. (for a full version see [6]).
6. Izuru Kitamura, Masanobu Katagi, and Tsuyoshi Takagi. A Complete Divisor Class Halving Algorithm for Hyperelliptic Curve Cryptosystems of Genus Two. *Cryptology ePrint Archive of IACR, Report 2004/255*, 2005.
7. Erik Woodward Knudsen. Elliptic Scalar Multiplication Using Point Halving. In *ASIACRYPT'99*, volume 1716 of *Lecture Notes in Computer Science*, pages 135–149. Springer-Verlag, 1999.
8. Tanja Lange. Formulae for Arithmetic on Genus 2 Hyperelliptic Curves. *Applicable Algebra in Engineering, Communication and Computing*, 15(5):295–328, 2005.
9. 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.
10. Rudolf Lidl and Harald Niederreiter. *Finite Fields*, volume 20 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, 1983.
11. Richard Schroeppel. Elliptic curve point halving wins big. 2nd Midwest Arithmetical Geometry in Cryptography Workshop, Urbana, Illinois, November 2000.

A Magma Implementation of the HLV Algorithm

Here is a sample Magma implementation of the HLV algorithm for the finite field $\mathbb{F}_{2^7}$.

```
GF2      := FiniteField(2);
R_GF2<z> := PolynomialRing(GF2);
F        := ext < GF2 | z^7 + z + 1 >;  // Define the extension field F_{2^7}
R<x>     := PolynomialRing(F);

f0 := F.1^41;                           // Setup the curve parameters
f2 := 1;                                //
f3 := F.1^32;                           //

f := x^5 + f3*x^3 + f2*x^2 + f0;        // Curve equation: y^2 + h(x)y = f(x)
h := x;                                 //

C := HyperellipticCurve(f, h);
J := Jacobian(C);
r := #J;                                // Compute the order of the Jacobian

halving := function(D)
    u0 := Coefficient(D[1], 0);
    u1 := Coefficient(D[1], 1);
    v0 := Coefficient(D[2], 0);
    v1 := Coefficient(D[2], 1);

    q1 := Sqrt(u1);
    q2 := 1 / q1;
    q3 := q2^2;
    q4 := u0 * q3;
    q5 := Sqrt(q2);
    q6 := Sqrt(q4);

    c  := u1 * (q6 + q5 + f3);
    xp := &+[c^2^(2*i + 1) : i in [0..2]];

    u1p := xp*q2;
    x   := u1p^2;
    s1  := v0 + (q1 + x + f3) * u0;
    u0p := Sqrt(s1);
    t   := Trace(u1p * (u0p + x + f3));

    if (t eq 0) then
        v0p := q5 * u0p + Sqrt(f0);
    else
        x   := x + q3;
        u1p := u1p + q2;
        u0p := u0p + q6;
        v0p := q5 * u0p + Sqrt(f0);
    end if;

    v1p := Sqrt(v1 + q1 * ((q1 + x + f3) * (x + f3) + u1) + f2);

    return [u1p, u0p, v1p, v0p];
end function;

// Now define a sample divisor class and compare D, 2D and the half of 2D
D := J ! [x^2 + F.1^18*x + F.1^80, F.1^17*x + F.1^117];
print "D: ", D;
print "2*D: ", 2*D;
halving(2*D);
```