

Faster Halvings in Genus 2

Peter Birkner¹ and Nicolas Thériault^{2*}

¹ Department of Mathematics and Computer Science, Coding Theory and Cryptology, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
`p.birkner@tue.nl`

² Instituto de Matemática y Física, Universidad de Talca, Casilla 747, Talca, Chile
`ntheriau@inst-mat.otalca.cl`

Abstract. We study divisor class halving for hyperelliptic curves of genus 2 over binary fields. We present explicit halving formulas for the most interesting curves (from a cryptographic perspective), as well as all other curves whose group order is not divisible by 4. Each type of curve is characterized by the degree and factorization form of the polynomial $h(x)$ in the curve equation. For each of these curves, we provide explicit halving formulæ for all possible divisor classes, and not only the most frequent case where the degree of the first polynomial in the Mumford representation is 2. In the optimal performance case, where $h(x) = x$, we also improve on the state-of-the-art and when $h(x)$ is irreducible of degree 2, we achieve significant savings over both the doubling as well as the previously fastest halving formulas.

Keywords. hyperelliptic curve, genus 2, halving, binary field.

1 Introduction

The double-and-add algorithm is essential to the efficiency of cryptosystems based on hyperelliptic curves. This algorithm (and many of its variations) is based on two basic group operations: the addition of two distinct group elements and the computation of the double of an element. An alternative that proved very successful in elliptic curves over binary fields is the halve-and-add algorithm, which relies on the computation of the “half” of a group element (of odd order), i.e. the computation of a pre-image of the doubling operation [9, 14]. Given the important savings produced by this approach for elliptic curves, it is natural to ask if similar results can be obtained for hyperelliptic curves over binary fields.

In a double-and-add algorithm, we can use explicit formulæ for the most common cases of the doubling and the addition, going back to Cantor’s polynomial-based algorithm if any special cases are encountered. In practice, using explicit formulæ for the special cases has no measurable impact on the average performance of a scalar multiplication, so it is not essential to develop them. The same

* The authors would like to thank the Fields Institute in Toronto for supporting this work. Research was partially supported by FONDECYT (Chile) grants #1070242 and by the Programa Reticulados y Ecuaciones of the Universidad de Talca.

is not true for the halve-and-add algorithm however, since we cannot easily describe the halving operation in terms of polynomial arithmetic. A halve-and-add algorithm should therefore contain explicit formulæ for all possible cases of the halving operation.

In this paper we investigate halving of divisor classes of hyperelliptic curves of genus 2 over finite fields of characteristic 2. Doubling formulæ for the different types of curves can be found in [10, 11, 1] and halving for some types of curves and/or cases have also been investigated in [7, 8, 3]. We present halving formulæ for all cases (most frequent and special cases) for all curves having at most one divisor of order 2 and no divisor of order 4 (see the second half of Section 2.2 for the reasons behind this condition). Rather than inverting the best doubling formulæ, we invert Cantor’s algorithm, using the divisibility condition on semi-reduced divisors to lower the cost. In the optimal performance case, where $h(x) = x$, we also improve on the state-of-the-art [3] and when $h(x)$ is irreducible of degree 2, we achieve significant savings over both the doubling [1] as well as the currently fastest halving formulas [7].

The remainder of this paper is structured as follows: Section 2 contains some important terminology and background. In Section 3, we develop a complete case study of the halving formulæ in the most efficient curves for cryptographic application, and in Section 4 we do the same for the most general type of curves where halving is of interest. For completeness, full sets of formulæ for the remaining types of curves where halving can be efficient are presented in the appendix, as well as the addition formula for the form of curve equation used in Section 4.

2 Basic Notations and Preliminaries

In this section we briefly recall the definitions of hyperelliptic curves, divisor class groups and the Mumford representation since we will use these notions throughout the whole paper.

A comprehensive resource for the mathematics of finite fields is [12]. 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 of genus 2 in characteristic 2). *Let \mathbb{F}_q be a field of characteristic 2 and $\overline{\mathbb{F}}_q$ its algebraic closure. A curve C , given by an equation of the form*

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

where $f \in \mathbb{F}_q[x]$ is a polynomial of degree 5 and $h \in \mathbb{F}_q[x]$ is a non-zero polynomial of degree at most 2, is called an imaginary hyperelliptic curve of genus 2 over \mathbb{F}_q if there is no point (x, y) on the curve over $\overline{\mathbb{F}}_q$ for which both partial derivatives are 0, i.e. such that $h(x) = 0$ and $f'(x) - h'(x)y = 0$ (This last condition ensures that the affine curve is non-singular).

Definition 2 (Divisor class group). *Given a hyperelliptic curve C of genus 2 over a binary field \mathbb{F}_q , the group of degree 0 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 2 over a binary field \mathbb{F}_q . Each nontrivial divisor class of C over \mathbb{F}_q can be represented by a unique pair of polynomials $u, v \in \mathbb{F}_q[x]$, where*

1. u is monic,
2. $\deg v < \deg u \leq 2$,
3. $u \mid v^2 + vh - f$.

A divisor satisfying Theorem 1 is called reduced (i.e. it is the shortest representative of its class), and if the condition $\deg(u) \leq 2$ is removed, the divisor is called semi-reduced. The divisibility condition will be essential in establishing some of the halving formulæ. Our halving formulæ expect the input divisor class to be in Mumford representation, work directly on the coefficients of the polynomials u and v and return an output in Mumford form. Since our goal is to compute pre-images of the group doubling, we refer to Algorithm 1 for a description of how this operation is performed using Cantor's algorithm.

Algorithm 1 Cantor's/Koblitz's doubling algorithm for genus 2 HEC in characteristic 2

INPUT: Reduced divisor $D = [u_a(x), v_a(x)]$
OUTPUT: Reduced divisor $D' = [u_c(x), v_c(x)]$, $D' = [2]D$

- 1: $d \leftarrow \gcd(u_a, h)$, $u_0 \leftarrow u_a/d$, $v_0 \leftarrow v_a \bmod u_0$
 - 2: $c \leftarrow h^{-1} \bmod u_0$, $u_1 \leftarrow u_0^2$, $v_1 \leftarrow v_0 + c(v_0^2 + v_0h + f) \bmod u_1$
 - 3: **if** $\deg(u_1) \leq 2$ **then**
 - 4: $u_c \leftarrow u_1$, $v_c \leftarrow v_1$
 - 5: **else**
 - 6: $u_c \leftarrow \text{monic} \left(\frac{f + v_1h + v_1^2}{u_1} \right)$, $v_c \leftarrow v_1 + h \bmod u_c$
 - 7: **end if**
 - 8: **return** $[u_c, v_c]$
-

We observe that all the pairs of polynomials computed in Algorithm 1 satisfy the divisibility condition of Theorem 1, i.e. $u_0 \mid v_0^2 + v_0h + f$, $u_1 \mid v_1^2 + v_1h + f$, and $u_c \mid v_c^2 + v_ch + f$. To obtain our halving formulæ, we sometime use the identities coming from these divisibility conditions rather than those which are more obvious in the polynomial equalities of Algorithm 1. In exchange, any identity which is not used to perform the halving becomes a divisibility condition that must be satisfied for $[u_i(x), v_i(x)]$ to be a semi-reduced divisor.

2.1 Field Arithmetic and Divisor Halving

Throughout this paper, we will assume that the field \mathbb{F}_q has order 2^n where n is not divisible by 2 or 3. This is mainly due to security concerns, since various versions of the Weil descent attack could be applied when n admits a factor of 2 or 3 (for example, see [6, 15, 5]). In fact, for cryptographic applications it is often assumed that n is a prime. As an added bonus, having n coprime to 2 means that we can take cube and 5-th roots in the field (since α^3 and α^5 are both isomorphism as 3 and 5 are coprime to $2^n - 1$), which allows us to simplify the curve equations a little more. Furthermore, since n will be odd we will have $\text{Tr}(1) = 1$. In various places, we implicitly take advantage of the identity $\text{Tr}(\alpha) = \text{Tr}(\alpha^2)$ to simplify some trace computations.

In finite fields of characteristic 2, some operations which are very expensive in fields of odd characteristic become very efficient, in particular computing the roots of a quadratic equation (when they are available in the field of definition). 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 [9, 14], and was recently extended to hyperelliptic curves of genus 2 [7, 8, 3]. In some fields, computing of square roots can be faster than the computation of squares [4, 2]. Two other operations, the trace (from \mathbb{F}_{2^n} to \mathbb{F}_2) and the half-trace (HT, to solve quadratic equations, see [4, 2]), can also be implemented to have similar costs to the squaring operation. For curves over those fields, it can be a good strategy to “replace” squares with square roots in the group arithmetic, which is often what halving does.

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

2.2 Choices of curves

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

$$y^2 + (h_2x^2 + h_1x + h_0)y = f_5x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0, \quad (2)$$

where $h(x) = h_2x^2 + h_1x + h_0 \neq 0$. It is also customary to use isomorphisms to impose that $f(x)$ is monic, i.e. that $f_5 = 1$, but we will relax this condition for some curves as the halving formulæ are more efficient if we use the isomorphisms to force $h_1 = 1$. We also note that when $h(x)$ is constant (i.e. $h_2 = h_1 = 0$), the curve is known to be supersingular, and therefore of limited interest for cryptography, but we will still cover these curves for completeness.

For the curve (2), the possible isomorphisms are given by $x = \alpha\tilde{x} + \beta$ and $y = \gamma\tilde{y} + \delta\tilde{x}^2 + \epsilon\tilde{x} + \zeta$, where both α and γ are nonzero, after which the equation is divided by γ^2 to get the coefficient of y^2 back to 1. We distinguish five types of curves depending on the degree and factorization type of $h(x)$:

- Ia $h_2 \neq 0$ and $h(x)$ irreducible: Using α and γ we can force $h_2 = h_1 = 1$. We can then use β to restrict $h(x)$ to $x^2 + x + 1$. The remaining freedom on β allows us to impose $\text{Tr}(f_5) \cdot \text{Tr}(f_4) = 0$. Taking advantage of δ , we can restrict f_4 to $\{0, 1\}$ and then ϵ and ζ allow us to remove f_3 and f_2 . We are left with $f(x) = f_5x^5 + f_4x^4 + f_1x + f_0$ where $f_4 \in \mathbb{F}_2$ and $f_4 \cdot \text{Tr}(f_5) = 0$.
- Ib $h_2 \neq 0$ and $h(x)$ is the product of two distinct linear factors: Using β and one of the roots of $h(x)$, we can force $h_0 = 0$. We can then use α and γ to restrict $h(x)$ to $x^2 + x$. The remaining freedom on β allows us to impose $\text{Tr}(f_5) \cdot \text{Tr}(f_4) = 0$. Taking advantage of δ , we can restrict f_4 to $\{0, 1\}$ and then ϵ and ζ allow us to remove f_3 and f_2 . We are left with $f(x) = f_5x^5 + f_4x^4 + f_1x + f_0$ where $f_4 \in \mathbb{F}_2$ and $f_4 \cdot \text{Tr}(f_5) = 0$.
- Ic $h_2 \neq 0$ and $h(x)$ a square: Using α and β and γ , we can force $h(x) = x^2$ and make $f(x)$ monic. ϵ and ζ allow us to remove f_3 and f_2 . Finally, δ can be used to limit f_4 to $\{0, 1\}$, leaving us with $f(x) = x^5 + f_4x^4 + f_1x + f_0$ with $f_4 \in \mathbb{F}_2$.
- II $h_2 = 0, h_1 \neq 0$: Using α and β and γ , we can force $h(x) = x$ and make $f(x)$ monic. δ and ζ allow us to remove f_4 and f_1 . Finally, we restrict f_2 using ϵ , giving us $f(x) = x^5 + f_3x^3 + f_2x^2 + f_0$ with $f_2 \in \mathbb{F}_2$.
- III $h_2 = h_1 = 0$: Using α and γ , we can force $h(x) = 1$ and make $f(x)$ monic. By selecting δ, ϵ and β wisely, we can remove f_4 and f_2 and reduce the number of possible values of f_1 (in general to a set of at most 16 values). Finally, ζ can be used to limit f_0 to $\{0, 1\}$, leaving us with $f(x) = x^5 + f_3x^3 + f_1x + f_0$ with $f_0 \in \mathbb{F}_2$.

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 $\frac{3}{2}q^3 + O(q^2)$ classes), followed by types II and Ic (each with $2q^2 + O(q)$ classes) and with type III (supersingular) the less common ($O(q)$ classes). From the point of view of the 2-torsion group, type Ic is closer to type II than type Ia and Ib.

We limit ourselves to curves for which the order of the Jacobian is either odd ($h(x)$ constant) or 2 times an odd number (which eliminates all type Ib curves). 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 pre-image of the doubling can always be computed and “becomes” unique (all other pre-images of the doubling have even order). The group order conditions are due to the following reasons:

1. To verify that the pre-image 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 divisors of order 2^r , we must make sure that we can halve the pre-image (at least) r times, which obviously affects the cost of our halving formulæ. When $r \geq 2$ (i.e. when there are divisors 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, there are four possible pre-images of the doubling. 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 formulæ 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 curves (other than type III), between a half and two thirds of them have divisors 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 [13], curves with divisors of order 4 are in general (slightly) weaker than those without, 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 formulæ 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.

3 Type II: $h(x) = x$

In this section, the curve C is of the form

$$y^2 + xy = x^5 + f_3x^3 + f_2x^2 + f_0, \quad f_2 \in \mathbb{F}_2.$$

Theorem 2. *Let $D_a = [u_a, v_a]$ be a divisor in Div_C^0 . If $\deg(u_a) = 2$, then D_a can be halved if and only if $\text{Tr}(u_{a1}(u_{a0} + f_3 + u_{a1}^2)) = 0$. If $\deg(u_a) = 1$, then D_a can be halved if and only if $\text{Tr}(f_2 + u_{a0}(u_{a0}^2 + f_3)) = 0$.*

Proof. To obtain these trace conditions, we try to invert the explicit formulæ for each of the possible cases of doubling (matching the outputs to the different forms of D_a). We do this by solving a sequence of equations to obtain a divisor D_c such that $[2]D_c = D_a$. We first observe that linear or square equations ($w^2 = \alpha$) pose no problem, whereas quadratic equations of the form $w^2 + w + \alpha = 0$ can be solved if and only if $\text{Tr}(\alpha) = 0$. If $\deg(u_a) = 1$, then we only encounter one quadratic (non-square) equation, with $\alpha = f_2 + u_{a0}(u_{a0}^2 + f_3)$. If $\deg(u_a) = 2$ and $u_{a1} = 0$ (which obviously satisfies $\text{Tr}(u_{a1}(u_{a0} + f_3 + u_{a1}^2)) = 0$), all equations are either linear or square. Finally, if $\deg(u_a) = 2$ and $u_{a1} \neq 0$, then we encounter one quadratic (non-square) equation, with $\alpha = u_{a1}(u_{a0} + f_3 + u_{a1}^2)$.

To show that the trace conditions are sufficient for the existence of a pre-image of the doubling, we show that D_c is indeed a valid divisor (i.e. the pair of polynomials computed correspond to a real divisor). In the computation of $D_c = [u_c(x), v_c(x)]$, we ignored a number of identities (essentially divisibility conditions) which are easily shown to be direct consequences of the divisibility conditions on D_a .

From Theorem 2, we obtain a simple condition for the group order:

Corollary 1. *The Jacobian of the curve C has order $2 \cdot \text{odd}$ if and only if $f_2 = 1$.*

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

Let D_{c_1} and D_{c_2} be the two pre-images of D_a , then $D_{c_1} - D_{c_2} = [x, \sqrt{f_0}] = D_{c_2} - D_{c_1}$. From this observation, we get the following corollary which allow us to distinguish the different special cases.

Corollary 2. *Let $D_a = [u_a(x), v_a(x)]$ be a divisor in Div_C^0 that can be halved and $D_c = [u_c(x), v_c(x)] = [\frac{1}{2}]D_a$ its pre-image (under the doubling) of odd order. If $\deg(u_a) = 2$ and $u_{a1} \neq 0$, then $\deg(u_c) = 2$ (“HLV22”). If $\deg(u_a) = 2$ and $u_{a1} = 0$, then $\deg(u_c) = 1$ or 2 (with $u_{c0} = 0$ in the second case) (“HLV21/22”). If $\deg(u_a) = 1$, then $\deg(u_c) = 2$ (“HLV12”).*

Finally, to verify that we are computing the pre-image of odd order, we use the conditions of Theorem 2 on $u_c(x)$ to ensure that it could be halved again (i.e. that it has odd order), and we correct the computations if necessary. We obtain the following formulæ:

Algorithm 2 (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 $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}] = [1/2]\overline{D}_a$

```

1:  $s_0 \leftarrow \sqrt{u_{a1}}, s_1 \leftarrow 1/s_0, s_2 \leftarrow s_1^2$  ▷ 1I+1S+1SR
2:  $s_3 \leftarrow \text{HT}(u_{a1}(u_{a0} + f_3 + s_0)), s_4 \leftarrow s_3s_2, s_5 \leftarrow s_4 + f_3$  ▷ 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  $\text{Tr}(s_5(s_7 + f_3^2 + u_{c1})) = 1$  then ▷ 1M+1TR
5:    $s_3 \leftarrow s_3 + 1, s_4 \leftarrow s_4 + s_2, s_5 \leftarrow s_5 + s_2$ 
6:    $s_6 \leftarrow s_6 + s_2u_{a0}, s_7 \leftarrow s_6, u_{c1} \leftarrow u_{c1} + s_1$  ▷ 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  $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$  ▷ 1I+8M+2S+4SR+1HT+1TR
11: ▷ Average: 1I+7.5M+2S+4SR+1HT+1TR

```

Algorithm 3 (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 $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}] = [1/2]\overline{D}_a$

```

1:  $s_0 \leftarrow \sqrt{u_{a0}}, s_1 \leftarrow f_3 + s_0, s_2 \leftarrow 1 + s_1u_{a0}$  ▷ 1M+1SR

```

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

Algorithm 4 (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 $[x + u_{c0}, v_{c0}] = [1/2]\overline{D}_a$
or $[x^2 + u_{c1}x, v_{c1}x + v_{c0}] = [1/2]\overline{D}_a$

1: $s_0 \leftarrow \sqrt{u_{a0}}, s_1 \leftarrow v_{a0} + s_0 v_{a1}$ ▷ 1M+1SR
2: **if** $\text{Tr}(u_{a0}s_0) = 1$ **then** ▷ 1M+1TR
3: $\overline{E} \leftarrow [x + s_0, s_1]$
4: **else**
5: $s_2 \leftarrow 1/s_0, v_{c1} \leftarrow (s_1 + \sqrt{f_0})s_2$ ▷ 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} ▷ 1I+4M+1SR+1TR
9: ▷ Average: 0.5I+3M+1SR+1TR

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

In this section, the curve C is of the form $y^2 + (x^2 + x + 1)y = f_5x^5 + f_4x^4 + f_1x + f_0$ with $f_4 \in \mathbb{F}_2$ and $f_4 \cdot \text{Tr}(f_5) = 0$. To improve the efficiency of the formulæ, we will assume that f_5^{-1} and f_5^{-2} are precomputed (only once per curve).

Theorem 3. *Let $D_a = [u_a(x), v_a(x)]$ be a divisor in Div_C^0 . If $\deg(u_a) = 2$, then D_a can be halved if and only if $\text{Tr}(u_{a1}f_5) = 0$. If $\deg(u_a) = 1$, then D_a can be halved if and only if $\text{Tr}(f_4 + f_5u_{a0}) = 0$.*

Proof. To prove this, we can follow the same ideas as in Theorem 2, with one difference: some of the halving formulæ require to solve two quadratic (non-square) equations rather than one. In those formulæ, it is easy to verify that switching between the roots of the first quadratic equation (adding one 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}_q) 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. *The Jacobian of the curve C has order 2·odd if and only if $\text{Tr}(f_5) = 1$ and $f_4 = 0$.*

Proof. The curve C has exactly one divisor of order 2, which is of the form $[x^2 + x + 1, v_h]$. The group order is divisible by 4 if and only if $[x^2 + x + 1, v_h]$ can be halved. From Theorem 3, this is possible if and only if $\text{Tr}(f_5) = 0$. The condition $f_4 = 0$ is then direct from the restriction $f_4 \cdot \text{Tr}(f_5) = 0$.

Let D_{c_1} and D_{c_2} the two pre-images of D_a , then their difference is $[x^2 + x + 1, v_h]$. From this observation, we get the following corollary which allow us to distinguish the different special cases:

Corollary 4. *Let $D_a = [u_a(x), v_a(x)]$ be a divisor in Div_C^0 that can be halved and $D_c = [u_c(x), v_c(x)] = [\frac{1}{2}]D_a$ its pre-image (under the doubling) of odd order. If $\deg(u_a) = 2$ and $u_{a1} \neq 0$, then $\deg(u_c) = 2$ (“HLV22”). If $\deg(u_a) = 2$ and $u_{a1} = 0$, then $\deg(u_c) = 1$ or 2 (“HLV21/22”). If $\deg(u_a) = 1$, then $\deg(u_c) = 2$ (“HLV12”).*

Finally, to verify that we are computing the pre-image of odd order, we use the conditions of Theorem 3 on $u_c(x)$ to ensure it could be halved again (i.e. that it has odd order), and we correct the computations if necessary. We obtain the following formulæ:

Algorithm 5 (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 $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}] = [1/2]\overline{D}$

```

1:  $s_0 \leftarrow f_5 u_{a1}$ ,  $s_1 \leftarrow u_{a1}^{-1}$ ,  $s_2 \leftarrow \text{HT}(s_0)$ ,  $s_3 \leftarrow s_2 s_1$  ▷ 1I+2M+1HT
2:  $s_4 \leftarrow s_2 u_{a1}$ ,  $s_5 \leftarrow f_5 u_{a0} + s_2 + s_3 + v_{a1} + 1 + u_{a1} + s_4$  ▷ 2M
3:  $s_6 \leftarrow s_5 u_{a1}$  ▷ 1M
4: if  $\text{Tr}(s_6) = 1$  then ▷ 1TR
5:    $s_2 \leftarrow s_2 + 1$ ,  $s_3 \leftarrow s_3 + s_1$ ,  $s_4 \leftarrow s_4 + u_{a1}$ 
6:    $s_5 \leftarrow s_5 + 1 + s_1 + u_{a1}$ ,  $s_6 \leftarrow s_5 u_{a1}$  ▷ 1M
7: end if
8:  $s_8 \leftarrow \text{HT}(s_6)$ ,  $s_9 \leftarrow s_8 s_1$  ▷ 1M+1HT

```

```

9:  $s_{10} \leftarrow s_3 u_{a0} + s_4 + s_8 + v_{a1} + 1 + u_{a1}$  ▷ 1M
10: if  $\text{Tr}((s_{10} + s_3 + s_9)(s_3 + f_5 + s_1)) = 1$  then ▷ 1M+1TR
11:    $s_8 \leftarrow s_8 + 1, s_9 \leftarrow s_9 + s_1, s_{10} \leftarrow s_{10} + 1$ 
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:  $\overline{E} \leftarrow [x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$ 
20: return  $\overline{E}$  ▷ 1I+16M+2SR+2TR+2HT
21: ▷ Average: 1I+15.5M+2SR+2TR+2HT

```

Note that for curves with $f_5 = 1$, the worst-case complexity 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 $\text{Tr}(s_{12})$), then the average complexity drops to 1I+12M+2SR+2HT+2TR (with that approach, a multiplication is required to correct s_{12}).

Algorithm 6 (HLV12, $h(x) = x^2 + x + 1, f(x) = f_5 x^5 + f_1 x + f_0$)

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

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

```

1:  $s_0 \leftarrow \text{HT}(f_5 u_{a0}), s_1 \leftarrow u_{a0}^2 + u_{a0}$  ▷ 1M+1S+1HT
2:  $s_2 \leftarrow v_{a0} + 1 + s_1(s_0 + 1) + s_0^2$  ▷ 1M+1S
3: if  $\text{Tr}(s_2) = 1$  then ▷ 1TR
4:    $s_0 \leftarrow s_0 + 1, s_2 \leftarrow s_2 + s_1 + 1$ 
5: end if
6:  $s_3 \leftarrow \text{HT}(s_2)$  ▷ 1HT
7: if  $\text{Tr}(s_3 f_5) = 1$  then  $s_3 \leftarrow s_3 + 1$  end if ▷ 1M+1TR
8:  $s_4 \leftarrow s_3 f_5^{-1}, s_5 \leftarrow s_0 + s_3, s_6 \leftarrow s_0^2 + s_3(1 + u_{a0} + s_3)$  ▷ 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:  $v_{c1} \leftarrow s_5 + s_0 u_{c1}, v_{c0} \leftarrow s_6 + s_0 u_{c0}$  ▷ 2M
11: return  $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$  ▷ 8M+3S+2SR+2TR+2HT

```

Algorithm 7 (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 $[x + u_{c0}, v_{a0}] = [1/2]\overline{D}_a$
or $[x^2 + u_{c1}x, v_{c1}x + v_{c0}] = [1/2]\overline{D}_a$

```

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

```

5 Conclusion

We investigated the halving of divisor classes of hyperelliptic curves of genus 2 over binary fields. We provided new and improved formulæ for all cases of the halving for all types of curves whose divisor class group has at most one divisor of order 2 and does not contain any divisor of order 4. We summarize our results for the most common cases in the table below, where we also compare with doubling formulæ [10, 11, 1] and with the fast halving formulæ available in previous papers [7, 8, 3].

Curve type	Doubling	Halving (previous best)	Halving (this work)
Ia (general)	1I+20M+6S	1I+19.5M+2S +2SR+2TR+2HT	1I+15.5M +2SR+2TR+2HT
Ia, with $h_1 = f_5 = 1$	1I+15M+7S	1I+13.5M+3S +2.5SR+2TR+2HT	1I+12M +2SR+2TR+2HT
Ic	1I+10M+6S	—	1I+10.5M +4SR+1TR+1HT
II	1I+5M+6S	1I+8M +5SR+1TR+1HT	1I+7.5M+2S +4SR+1TR+1HT
III	1I+4M+6S	—	1I+4M+6SR

Note that Kitamura, Katagi and Takagi [7, 8] also provide a brief description of halving for type Ic curves, but the best approximation for the cost would be from their formulæ for type Ia curves with $h_1 = f_5 = 1$.

For curves of type Ic and III, our halving formulæ are as efficient as the doubling. For type II curves, our halving formulæ improve on the state-of-the-art, although not sufficiently to match the efficiency of the doubling. Our most important gain comes from type Ia curves, where our halving cost is significantly lower than both the doubling and the previous best halving formulæ.

References

1. Roberto M. Avanzi, Henri Cohen, Christophe Doche, Gerhard Frey, Tanja Lange, Kim Nguyen and Frederik Vercauteren: Handbook of Elliptic and Hyperelliptic Curve Cryptography. Chapman & Hall/CRC, 2006.
2. Roberto M. Avanzi: A Note on Square Roots in Binary Fields, preprint.
3. Peter Birkner: Efficient Divisor Class Halving on Genus Two Curves. In: Selected Areas in Cryptography - SAC 2006, Lecture Notes in Computer Science, Vol. 4356 (2007), pp. 317–326.
4. Kenny Fong, Darrel Hankerson, Julio López and Alfred Menezes: Field Inversion and Point Halving Revisited. IEE Trans. Computers, Vol. 53 No. 8 (2004), pp. 1047–1059.
5. Pierrick Gaudry: Index calculus for abelian varieties and the elliptic curve discrete logarithm problem, preprint (2004), available at <http://eprint.iacr.org/2004/073/>.
6. Pierrick Gaudry, Florian Hess and Nigel P. Smart: Constructive and destructive facets of Weil descent on elliptic curves. Journal of Cryptology, Vol. 15, No. 1 (2002), pp. 19–46.
7. Izuru Kitamura, Masanobu Katagi, and Tsuyoshi Takagi: A Complete Divisor Class Halving Algorithm for Hyperelliptic Curve Cryptosystems of Genus Two, preprint (2005), available at <http://eprint.iacr.org/2005/255/>.
8. Izuru Kitamura, Masanobu Katagi, and Tsuyoshi Takagi: A Complete Divisor Class Halving Algorithm for Hyperelliptic Curve Cryptosystems of Genus Two. Information Security and Privacy ACISP 2005, Lecture Notes in Computer Science, Vol. 3574 (2005), pp. 146–157. (for a full version see [7]).
9. Erik Woodward Knudsen: Elliptic scalar multiplication using point halving. In: Advances in Cryptology – Asiacrypt 1999, Lecture Notes in Computer Science, Vol. 1716 (1999), pp. 135–149.
10. Tanja Lange: Formulae for Arithmetic on Genus 2 Hyperelliptic Curves. Applicable Algebra in Engineering, Communication and Computing, Vol. 15, No. 5 (2005), pp. 295–328.
11. Tanja Lange and Marc Stevens: Efficient Doubling for Genus Two Curves over Binary Field. Selected Areas in Cryptography SAC 2004, Lecture Notes in Computer Science, Vol. 3357 (2005), pp. 170–181.
12. Rudolf Lidl and Harald Niederreiter: Finite Fields, 2nd ed., Cambridge University Press, 1997.
13. Stephen Pohlig and Martin Hellman: An improved algorithm for computing logarithms over $\text{GF}(p)$ and its cryptographic significance. IEEE Trans. Inform. Theory, Vol. IT-24 (1978), pp. 106–110.
14. Richard Schroeppe: Elliptic curves: Twice as fast!. Crypto 2000 Rump Session.
15. Nicolas Thériault: Weil Descent for Artin-Schreier Curves, preprint (2003), available at <http://homepage.mac.com/ntheriau>.

A Type III: $h(x) = 1$

In this section, the curve C is of the form $y^2 + y = x^5 + f_3x^3 + f_1x + f_0$ with $f_0 \in \mathbb{F}_2$. Doing a case-by-case study of the doubling algorithm gives us the following theorem, and the halving formulæ are obtained by inverting the doubling ones:

Theorem 4. *Let $D_a = [u_a(x), v_a(x)]$ be a reduced divisor in Div_C^0 and $D_c = [u_c(x), v_c(x)] = \frac{1}{2}D_a$ its pre-image under the doubling. If $\deg(u_a) = 2$, then $\deg(u_c) = 2$ if and only if $u_{a1} \neq 0$ (HLV22), otherwise $\deg(u_c) = 1$ (HLV21). If $\deg(u_a) = 1$, then $\deg(u_c) = 2$ (HLV12).*

Algorithm 8 (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 $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}] = [1/2]\overline{D}$

- 1: $s_0 \leftarrow \sqrt{u_{a1}}, s_1 \leftarrow 1/s_0, s_2 \leftarrow s_1 + f_3, u_{c1} \leftarrow \sqrt{s_2}$ ▷ 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 9 (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 $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}] = [1/2]\overline{D}$

- 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 10 (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 $[x + u_{c0}, v_{c0}] = [1/2]\overline{D}_a$

- 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
-

B Type Ic: $h(x) = x^2$

In this section, the curve C is of the form $y^2 + x^2y = x^5 + f_4x^4 + f_1x + f_0$ with $f_4 \in \mathbb{F}_2$.

Theorem 5. *Let $D_a = [u_a(x), v_a(x)]$ be a divisor in Div_C^0 . If $\deg(u_a) = 2$, then D_a can be halved if and only if $\text{Tr}(u_{a1}) = 0$. If $\deg(u_a) = 1$, then D_a can be halved if and only if $\text{Tr}(f_4 + u_{a0}) = 0$.*

Proof. As in Theorem 2.

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

Corollary 5. *The Jacobian of the curve C has order $2 \cdot \text{odd}$ if and only if $f_4 = 1$.*

Proof. The curve C has exactly one divisor of order 2, which is of the form $[x, \sqrt{f_0}]$. The group order is divisible by 4 if and only if $[x, \sqrt{f_0}]$ can be halved. From Theorem 5, this is possible if and only if $\text{Tr}(f_4) = 0$ and since $f_4 \in \mathbb{F}_2$, C has a divisor of order 4 if and only if $f_4 = 0$.

Let D_{c_1} and D_{c_2} the two pre-images of D_a , then $D_{c_1} - D_{c_2} = D_{c_2} - D_{c_1} = [x, \sqrt{f_0}]$. From this observation, we get the following corollary which allow us to distinguish the different special cases.

Corollary 6. *Let $D_a = [u_a(x), v_a(x)]$ be a divisor in Div_C^0 that can be halved and $D_c = [u_c(x), v_c(x)] = \frac{1}{2}D_a$ its pre-image (under the doubling) of odd order. If $\deg(u_a) = 2$ and $u_{a1} \neq 0$, then $\deg(u_c) = 2$ (“HLV22”). If $\deg(u_a) = 2$ and $u_{a1} = 0$, then $\deg(u_c) = 1$ or 2 (with $u_{c0} = 0$ in the second case) (“HLV21/22”). If $\deg(u_a) = 1$, then $\deg(u_c) = 2$ (“HLV12”).*

Finally, to verify that we are computing the pre-image of odd order, we use the conditions of Theorem 5 on $u_c(x)$ to ensure that it could be halved again (i.e. that it has odd order), and we correct the computations if necessary. We obtain the following formulæ:

Algorithm 11 (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 $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}] = [1/2]\overline{D}$

```

1:  $s_0 \leftarrow 1/u_{a1}$ ,  $s_1 \leftarrow \text{HT}(u_{a1})$ ,  $s_2 \leftarrow s_1 s_0$                                 ▷ 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:  $s_5 \leftarrow v_{a1} + u_{a1}(1 + s_4 + s_1) + s_2 u_{a0}$ ,  $s_6 \leftarrow s_5(s_2 + 1 + s_0)$         ▷ 3M
4: if  $\text{Tr}(s_6) = 1$  then                                                                    ▷ 1TR
5:    $s_2 \leftarrow s_2 + s_0$ ,  $s_3 \leftarrow s_3 + f_1 s_0$ ,  $s_4 \leftarrow s_4 + 1$                 ▷ 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: $u_{c1} \leftarrow \sqrt{s_6}, u_{c0} \leftarrow \sqrt{s_3}, v_{c0} \leftarrow \sqrt{f_0 + s_3(1 + s_4)}$ \triangleright 1M+3SR
9: $v_{c1} \leftarrow s_6 + s_2u_{c0} + s_4u_{c1}$ \triangleright 2M
10: **return** $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$ \triangleright 1I+12M+4SR+1HT+1TR
11: \triangleright Average: 1I+10.5M+4SR+1HT+1TR

Algorithm 12 (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 $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}] = [1/2]\overline{D}_a$

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

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 $[x + u_{c0}, v_{c0}] = [1/2]\overline{D}_a$
or $[x^2 + u_{c1}x, v_{c1}x + v_{c0}] = [1/2]\overline{D}_a$

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

C Genus 2 addition for type Ia, $h(x) = x^2 + x + 1$

The form of the equation we used for curves of type Ia differs from the one more commonly used in explicit formulæ paper (which prefer to force $f_5 = 1$ rather than $h_1 = 1$). It is therefore natural to ask what impact the form of the equation has on the group addition, which is also necessary for the halving-and-add algorithm.

As the following formula shows, allowing $f_5 \neq 1$ allows us to perform the group addition in 1I+21M+4S, which is in fact slightly more efficient than the 1I+22M+3S required with the more common form of the equation.

Algorithm 14 (ADD22, $h(x) = x^2 + x + 1$, $f(x) = f_5x^5 + f_1x + f_0$)

INPUT: The divisor classes $\overline{D}_a = [x^2 + u_{a1}x + u_{a0}, v_{c1}x + v_{a0}]$,
 $\overline{D}_b = [x^2 + u_{b1}x + u_{b0}, v_{b1}x + v_{b0}]$

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

```

1:  $s_0 \leftarrow u_{a1} + u_{b1}$ ,  $s_1 \leftarrow u_{a1}s_0$ ,  $s_2 \leftarrow u_{a0} + u_{b0}$ ,  $s_3 \leftarrow s_1 + s_2$  ▷ 1M
2:  $s_4 \leftarrow s_3s_2 + u_{a0}s_0^2$ ,  $s_5 \leftarrow v_{a0} + v_{b0}$ ,  $s_6 \leftarrow v_{a1} + v_{b1}$  ▷ 2M+1S
3:  $s_7 \leftarrow s_3s_5$ ,  $s_8 \leftarrow s_0s_6$  ▷ 2M
4:  $s_9 \leftarrow (s_3 + s_0)(s_5 + s_6) + s_7 + s_8(1 + u_{a1})$ ,  $s_{10} \leftarrow s_4s_9$  ▷ 3M
5: if  $s_{10} = 0$  then
6:   Use Cantor's algorithm
7: else
8:    $s_{11} \leftarrow s_7 + u_{a0}s_8$ ,  $s_{12} \leftarrow 1/s_{10}$ ,  $s_{13} \leftarrow s_4s_{12}$  ▷ 1I+2M
9:    $s_{14} \leftarrow s_9^2s_{12}$ ,  $s_{15} \leftarrow s_4s_{13}$ ,  $s_{16} \leftarrow s_{15}^2$ ,  $s_{17} \leftarrow s_{11}s_{13}$  ▷ 3M+2S
10:   $s_{18} \leftarrow f_5s_{16}$ ,  $u_{c1} \leftarrow s_0 + s_{15} + s_{18}$  ▷ 1M
11:   $u_{c0} \leftarrow s_2 + s_{17}^2 + s_1 + (1 + s_{17} + u_{a1})s_{15} + s_0s_{18}$  ▷ 2M+1S
12:   $s_{19} \leftarrow u_{b1} + u_{c1}$ ,  $s_{20} \leftarrow (s_{17} + u_{b1})s_{19}$  ▷ 1M
13:   $s_{21} \leftarrow (u_{c0} + u_{b0})s_{17} + u_{b0}s_{19}$ ,  $v_{c1} \leftarrow u_{c1} + 1 + v_{b1} + s_{14}s_{20}$  ▷ 3M
14:   $v_{c0} \leftarrow u_{c0} + 1 + v_{b0} + s_{14}s_{21}$  ▷ 1M
15: end if
16: return  $[x^2 + u_{c1}x + u_{c0}, v_{c1}x + v_{c0}]$  ▷ 1I+21M+4S

```
