Journal of Cryptographic Engineering
https://doi.org/10.1007/s13389-024-00351-7

RESEARCH ARTICLE

Complete group law for genus 2 Jacobians on Jacobian coordinates

Elif Ozbay Gurler[1] · **Huseyin Hisil**[2]

Received: 16 February 2023 / Accepted: 9 March 2024

Abstract
This manuscript provides complete, inversion-free, and explicit group law formulas in Jacobian coordinates for the genus 2 hyperelliptic curves of the form $y^2 = x^5 + a_3x^3 + a_2x^2 + a_1x + a_0$ over a field K with $\mathrm{char}(K) \neq 2$. The formulas do not require the use of polynomial arithmetic operations such as resultant, mod, or gcd computations but only operations in K.

Keywords Group law · Genus 2 · Hyperelliptic curves · Explicit formulas · Jacobian coordinates

1 Introduction

Elliptic curves are offered for the use of cryptographic applications by Miller in 1985 [1] and then by Koblitz in 1987 [2]. Two years later, Koblitz [3] proposed that hyperelliptic curves of arbitrary genus can replace elliptic curves in cryptosystems. However, this claim was falsified by Gaudry [4], who showed that security is conversely proportional to the genus. Gaudry's work led the cryptographic studies to be focused on the genus 1 and genus 2 cases specifically.

A genus 1 curve is an elliptic curve. The points on an elliptic curve can be made into an abelian group. In comparison, a hyperelliptic curve of genus ≥ 2 is not an algebraic group by itself; however, its Jacobian becomes an abelian group under the binary operation divisor addition. A landmarking algorithm for divisor addition was introduced by David G. Cantor in 1987 [5, §3, §4]. Cantor's algorithm is deterministic and works for arbitrary genus hyperelliptic curves. The algorithm operates on Mumford's coordinates [6], a polynomial representation of divisors, and, consequently, makes heavy use of polynomial arithmetic, which makes it relatively inefficient for cryptographic applications in its original form.

Cantor's algorithm is improved by others [3, 7–9], and explicit formulas for hyperelliptic curves of particular genus are studied. Lange reports in her thesis [10] that Spallek [11] proposed explicit formulas for the most common addition on genus 2 curves, later optimized by Harley [12] for odd characteristic curves. Lange [10] advanced Harley's approach to arbitrary characteristics. In her thesis [10] and her following work [13], Lange presented a complete divisor addition algorithm on genus 2 hyperelliptic curves in a semi-explicit manner. Speed-ups [14–16] followed Lange's work. The complete inversion-free projective formulas are proposed in [17–19]. In 2011, Costello and Lauter [20] derived new formulas for Cantor's composition step, which incorporates solving linear systems instead of polynomial arithmetic. In 2014, Hisil and Costello [21] extended the work in [20] by introducing Jacobian coordinates, reducing the operation counts significantly. A recent work by Hu et al. [22] further extends the work in [21] to the case of degenerate divisor addition. Apart from these developments, which are the main focus of this work, Gaudry [23] provided explicit formulas to perform pseudo-group operations on genus 2 Kummer surfaces which are currently the speed leader in curve-based cryptosystems; see the implementation in [24]. However, some cryptographic constructions enforce the use of prime order genus 2 Jacobians, which do not allow a Kummer parameterization.

This work aims to present complete, inversion-free, and explicit formulas for divisor addition on genus 2 hyperelliptic curves. We closely follow Lange's thesis [10] and present an algorithm in affine coordinates. We dispose of all polynomial arithmetic by expressing the operations explicitly. As the main contribution of this paper, we propose the formulas in Jacobian coordinates by exploiting [21]. The formulas

✉ Elif Ozbay Gurler
elif.oezbay@tu-darmstadt.de

✉ Huseyin Hisil
hhisil@uow.edu.au

[1] Computer Engineering Department, Yaşar University, Üniversite caddesi, 35100 İzmir, Türkiye

[2] School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia

Published online: 16 April 2024

combine the works in [21] and [22] and handle the missing cases for the sake of a complete inversion-free algorithm.

The paper is organized as follows. We briefly cover the fundamental background in Sect. 2. In Sect. 3, we present the complete and explicit formulas in affine coordinates for divisor addition as an algorithm. We present the inversion-free formulas in Jacobian coordinates for the missing cases in the literature; see Sect. 4. The cost analysis is given in Sect. 5. We derive our conclusions in Sect. 6. For the convenience of the reader, we attach computer-aided verification scripts in the Appendix.

2 Preliminaries

This section recalls basic definitions in the theory of hyperelliptic curves. Throughout this section, we assume that K is a field with char(K) $\neq 2$.

2.1 Hyperelliptic curves

A hyperelliptic curve H of genus g over K is a non-singular curve with the equation

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

where $h(x), f(x) \in K[x]$, $\deg(h) \leq g$, $\deg(f) = 2g + 1$, and $f(x)$ is monic. We denote the point set of H as $H(\bar{K}) = \{(x, y) \in \bar{K} \times \bar{K} : y^2 + h(x)y = f(x)\} \cup \{\mathcal{O}\}$ where $\mathcal{O}$ is the single point at infinity. Let $P = (x, y)$, the negative of P is given as $-P = (x, -y - h(x))$. In the following sections, we will use the notation $f'(x)$ and $f''(x)$ for the derivatives $\frac{df(x)}{dx}$ and $\frac{d^2 f(x)}{dx^2}$, respectively.

Under the assumption char$(K) \neq 2$, H can be put in the simplified form $y^2 = f(x)$. Additionally, under the assumption char$(K) \neq 5$, a genus 2 hyperelliptic curve can be put in the form

$$y^2 = x^5 + a_3x^3 + a_2x^2 + a_1x + a_0.$$

We use the letter H in the following sections to address such a curve. We note here that the formulas presented in this work apply to char$(K) = 5$ provided that $\Delta_H \neq 0$, where Δ is the discriminant of $f(x)$.

2.2 Divisors

A divisor D on H is a formal sum

$$D = \sum_{P \in H(\bar{K})} n_P(P)$$

where only a finite number of $n_P \in \mathbb{Z}$ are nonzero. The identity is called the zero divisor, denoted 0, with $\forall P,\ n_P = 0$. The support of D is the set $\text{supp}(D) = \{P \in H(\bar{\mathbb{K}}) : n_P \neq 0\}$ and the degree of D is given by $\deg(D) = \sum_{P \in H(\bar{K})} n_P$. The order of D at a point $P \in H(\bar{\mathbb{K}})$ is the coefficient n_P. A divisor of a function f is given by $(f) = \sum_{P \in H(\bar{K})} \text{ord}_f(P)$ where $\text{ord}_f(P)$ is the multiplicity of f on P.

A semi-reduced divisor has the form $D = \sum n_P(P) - (\sum n_P)(\mathcal{O})$ where $n_P \geq 0$ and $-P \notin \text{supp}(D)$ if $P \in \text{supp}(D)$ unless $P = -P$ with $n_P = 1$. The inverse of a semi-reduced divisor is formulated as $-D = \sum n_P(-P) - (\sum n_P)(\mathcal{O})$. A reduced divisor is a semi-reduced divisor with $\sum n_P \leq g$.

Throughout this paper we will refer to divisors with one and two points in their support (other than $\mathcal{O}$) as; degenerate and non-degenerate divisors, respectively.

On H, the set of all divisors, denoted $\text{Div}_{\bar{K}}(H)$; degree zero divisors, namely $\text{Div}^0_{\bar{K}}(H)$; the principal divisors (divisors of a function), denoted $\text{Prin}_{\bar{K}}(H)$ form a group. Moreover, the groups mentioned satisfy

$$\text{Prin}_{\bar{K}}(H) \subseteq \text{Div}^0_{\bar{K}}(H) \subseteq \text{Div}_{\bar{K}}(H).$$

The quotient group $\text{Div}^0_{\bar{K}}(H)/\text{Prin}_{\bar{K}}(H)$ is called the divisor class group, which is isomorphic to the Jacobian, denoted by $J(H)$. Two divisors $D_1, D_2 \in J(H)$ are equivalent, denoted $D_1 \sim D_2$, if $D_1 - D_2 \in \text{Prin}_{\bar{K}}(H)$. An important observation about the Jacobian is that each coset of it has a unique reduced divisor. Moreover, let $D_1, D_2 \in J(H)$ be reduced divisors; then there exists a reduced divisor D such that $D \sim D_1 + D_2$.

A reduced divisor $D \in J(H)$ can be represented using the Mumford representation [6] with two polynomials as $D = [u(x), v(x)]$ where the following assumptions hold

(i) u is monic,
(ii) $\deg(v) < \deg(u) \leq 2$,
(iii) $u \mid v^2 - f$.

Mumford representation of a degenerate divisor $D = (P_1) - (\mathcal{O})$ where $P_1 = (x_1, y_1) \in H(K)$ has the form $D = [x - x_1, y_1]$. Notice that the point in the support of a degenerate divisor is defined to be in $H(K)$. For a non-degenerate divisor $D = (P_1) + (P_2) - 2(\mathcal{O})$ where $P_1 = (x_1, y_1) \in H(\bar{K})$ and $P_2 = (x_2, y_2) \in H(\bar{K})$, we will use the notation $D = [x^2 + qx + r, sx + t]$ with $q, r, s, t \in K$. Here, assuming $x_1 \neq x_2$,

$$\begin{aligned} q &= -x_1 - x_2, \\ r &= x_1x_2, \\ s &= \frac{y_1 - y_2}{x_1 - x_2}, \\ t &= \frac{x_1y_2 - x_2y_1}{x_1 - x_2}, \end{aligned} \tag{2}$$

and otherwise

$$
\begin{aligned}
q &= -2x_1, \\
r &= x_1^2, \\
s &= \frac{5x_1^4 + 3a_3x_1^2 + 2a_2x_1 + a_1}{2y_1}, \\
t &= \frac{-3x_1^5 - a_3x_1^3 + a_1x_1 + 2a_0}{2y_1}.
\end{aligned} \tag{3}
$$

We note that $y_1 = 0$ is not possible by definition of a reduced divisor whose support cannot contain opposite points. Finally, the identity is represented by the divisor $[1, 0]$.

2.3 Cantor's algorithm

In 1987, Cantor introduced an algorithm [5, §3, §4] to compute divisor addition on hyperelliptic curves, analogous to class group arithmetic, which is generalized and presented later by [3, 25]. Cantor's algorithm uses polynomial arithmetic and works for all hyperelliptic curves of arbitrary genus. There are two stages of the algorithm; composition stage (Algorithm 1 lines 1–5) computes the semi-reduced divisor that is equal to the addition of the inputs; reduction stage (Algorithm 1 lines 6–12) computes the related unique reduced divisor. For further information, we refer the reader to the appendix of [26]. We use this algorithm to verify the correctness of the explicit formulas presented in Sect. 3. Since the computations involving these verifications are tedious, we prefer to carry them out succinctly with the help of computer algebra rather than pages-long algebraic investigation. We now present the complete and explicit group law in Sect. 3.

Algorithm 1 Cantor's Algorithm [3, 5, 25]

Require: Reduced divisors $D_1 = [u_1, v_1]$, $D_2 = [u_2, v_2]$ such that $D_1, D_2 \in J(H)$
Ensure: Reduced divisor $D_3 = [u, v]$ where $D_3 \sim D_1 + D_2$
1: Compute d_1, e_1, e_2 where $d_1 = \gcd(u_1, u_2) = e_1u_1 + e_2u_2$
2: Compute d, c_1, c_2 where $d = \gcd(d_1, v_1+v_2+h) = c_1d_1+c_2(v_1+v_2+h)$
3: Let $s_1 \leftarrow c_1e_1$, $s_2 \leftarrow c_1e_2$, $s_3 \leftarrow c_2$ such that $d = s_1u_1 + s_2u_2 + s_3(v_1 + v_2 + h)$
4: $u \leftarrow \dfrac{u_1u_2}{d^2}$
5: $v \leftarrow \dfrac{s_1u_1v_2 + s_2u_2v_1 + s_3(v_1v_2 + f)}{d} \mod u$
6: $u' \leftarrow \dfrac{f - vh - v^2}{u}$
7: $v' \leftarrow (-h - v) \mod u'$
8: **if** $\deg(u') > g$ **then**
9: $u \leftarrow u', v \leftarrow v'$
10: Go to step 6
11: **end if**
12: $u' \leftarrow \text{monic}(u')$
13: **return** u', v'

3 Explicit formulas for divisor addition

We start with basic facts which are referenced in the body of our algorithm.

Lemma 1 *Let D_1 and D_2 be degenerate divisors such that $D_1 \neq -D_2$. Then, $D_1 + D_2$ is non-degenerate.*

Proof This is simply Mumford's composition. □

Lemma 2 *Let D_1 be a degenerate and D_2 be a non-degenerate divisor. Then, $D_1 + D_2$ cannot be the zero divisor.*

Proof Let $D_1 = [x - x_1, y_1]$ and $D_2 = [x^2 + q_2x + r_2, s_2x + t_2]$ as in the definition of the lemma. $-D_1 = [x - x_1, -y_1]$ is degenerate by construction. Assume that $D_1 + D_2 = 0$. So, $D_2 = -D_1$. However, D_2 is non-degenerate, contradiction. Thus, the addition of a degenerate divisor with a non-degenerate divisor cannot give the zero divisor. □

Lemma 3 *There exists no degenerate divisor $D \in J(H)$ such that $3D = 0$.*

Proof Let D be a divisor in $J(H)$ such that $3D = 0$. Assume that $D = (P) - (\mathcal{O})$ is degenerate. So, $2D = (-P) - (\mathcal{O})$. Nevertheless, by Lemma 1, $2D$ must be non-degenerate while $(-P) - (\mathcal{O})$ is clearly degenerate. This is a contradiction by Lemma 2. Therefore, D must be non-degenerate. □

In the remainder of this section, we define two input divisors D_1, D_2, and an output divisor $D_3 = D_1 + D_2$, and examine the complete formulas for D_3. We set divisor $D_3 = (P_5) - (\mathcal{O}) = [x - x_5, y_5]$ if D_3 is degenerate and $D_3 = (P_5) + (P_6) - 2(\mathcal{O}) = [x^2 + q_3x + r_3, s_3x + t_3]$ if non-degenerate.

3.1 Trivial inputs

Here, we investigate the output when at least one of the input divisors is the identity, namely the zero divisor. The cases are trivial, i.e., if one input divisor is 0 then the output is the other input divisor (which is also true when both inputs are 0).

3.2 Degenerate/degenerate inputs

Let $D_1 = (P_1) - (\mathcal{O}) = [x - x_1, y_1]$ and $D_2 = (P_3) - (\mathcal{O}) = [x - x_3, y_3]$. Now, we investigate whether D_1 and D_2 are joint by examining the points at their support: $P_1 = (x_1, y_1)$ and $P_3 = (x_3, y_3)$ in $H(K)$.

- Case $x_1 = x_3$: The input divisors are joint with $P_1 = \pm P_3$.
  - Case $y_1 = -y_3$: We have $P_1 = -P_3$. Thus, $D_1 = -D_2$ and $D_3 = 0$.
  - Case $y_1 = y_3$: We have $P_1 = P_3$. Therefore, $D_1 = D_2$ and $D_3 = 2D_1 = 2(P_1) - 2(\mathcal{O}) = [x^2 + q_3x + r_3,\ s_3x + t_3]$ is non-degenerate and can be computed with Mumford's composition (3). [Script 2] in the Appendix verifies the correctness of this formula.
- Case $x_1 \neq x_3$: The input divisors are disjoint, and $D_3 = [x^2+q_3x+r_3,\ s_3x+t_3]$ can be computed with Mumford's composition (2). [Script 3] in the Appendix verifies the correctness of this formula.

Since the conditions above exhaust all possible cases, a degenerate output is not possible, as stated in Lemma 1. Note that the case where $x_1 = x_3$ and $y_1 = y_3 = 0$ (special point doubling) is handled by "Case $y_1 = -y_3$".

3.3 Degenerate/non-degenerate inputs

We continue with the addition of a non-degenerate and a degenerate divisor. We assume that the inputs are swapped, if necessary, in order to have D_1 and D_2, degenerate and non-degenerate, respectively. Let $D_1 = (P_1)-(\mathcal{O}) = [x-x_1, y_1]$ and $D_2 = (P_3) + (P_4) - 2(\mathcal{O}) = [u_2(x), v_2(x)] = [x^2 + q_2x+r_2,\ s_2x+t_2]$ be divisors in $J(H)$ where $P_1 = (x_1, y_1) \in H(K) - \{\mathcal{O}\}$.

- Case $u_2(x_1) = 0$: Since $x_1 \in K$ is a root of the polynomial $u_2(x)$, we deduce that $u_2(x)$ must have the roots $x_3, x_4 \in K$ satisfying $(x_3 - x_1)(x_4 - x_1) = 0$, though we do not know x_3 and x_4 explicitly. Without loss of generality, we assume $x_3 = x_1$. Now, we compute, $y_3 = v_2(x_1)$, $x_4 = -q_2 - x_3$, and $y_4 = v_2(x_4)$. Clearly, $x_1, y_1, x_3, y_3, x_4, y_4 \in K$. Before we proceed to investigate the following subcases, we note that $P_3 = (x_3, y_3)$ and $P_4 = (x_4, y_4)$ are in $H(K) - \{\mathcal{O}\}$.
  - Case $y_1 = -y_3$: We have $P_1 = -P_3$. The points P_1 and P_3 cancel out each other which results in the degenerate output $D_3 = (P_4) - (\mathcal{O}) = [x - x_4, y_4]$.
  - Case $y_1 = y_3$: Here $P_1 = P_3$ and we continue with investigating the rare case $P_1 = P_3 = P_4$.
    * Case $x_1 = x_4$: We have $P_1 = \pm P_4$. Now, $P_1 = -P_4$ is not possible, because otherwise $D_2 = (P_1) + (-P_1) - 2(\mathcal{O})$ would be non-reduced. Thus, $P_1 = P_3 = P_4$ which leads us to the formula $D_3 = 3(P_1) - 3(\mathcal{O}) = 3D_1$ in [22, §3.2] with

$$\begin{aligned} q_3 &= 3x_1 - A^2, \\ r_3 &= a_3 - 2AB + 3x_1(q_3 - x_1), \\ s_3 &= Aq_3 - B, \\ t_3 &= Ar_3 - C \end{aligned} \tag{4}$$

where

$$\begin{aligned} A &= \frac{2y_1^2 f''(x_1) - f'(x_1)^2}{8y_1^3}, \\ B &= \frac{f'(x_1)}{2y_1} - 2Ax_1, \\ C &= y_1 - Ax_1^2 - Bx_1. \end{aligned}$$

Note that $D_1 = (P_1) - (\mathcal{O})$ cannot have order three[1]. Note also that the denominators of q_3, r_3, s_3, and t_3 are all powers of y_1. Here, $y_1 \neq 0$ because the case $y_1 = 0$ is handled in "Case $y_1 = -y_3$". Therefore, the denominators never vanish. [Script 4] in the Appendix verifies the correctness of this formula.

    * Case $x_1 \neq x_4$: Now, P_4 is disjoint with other points in both supports. So, the addition consists of the doubling of $D_1 = (P_1) - (\mathcal{O})$ and the accumulation of $(P_4) - (\mathcal{O})$ on $2D_1$. In this case, we compute divisor $2(P_1) - 2(\mathcal{O}) = [x^2 + qx+r,\ sx+t]$ as in Eq. (3). Note that the denominator y_1 in Eq. (3) does not vanish since the case $y_1 = 0$ is again handled in "Case $y_1 = -y_3$". Now, $D_3 = 2(P_1) + (P_4) - 3(\mathcal{O})$ can be computed with the generic formula in [22, §3.1] as follows,

$$\begin{aligned} q_3 &= x_4 - q - A^2, \\ r_3 &= a_3 + q^2 - r - A(B + s) \\ &\quad + x_4q_3, \\ s_3 &= Aq_3 - B, \\ t_3 &= Ar_3 - C \end{aligned} \tag{5}$$

where

$$\begin{aligned} A &= \frac{y_4 - sx_4 - t}{x_4^2 + qx_4 + r}, \\ B &= s + qA, \\ C &= t + rA. \end{aligned}$$

Notice that the denominator of A vanishes only when P_4 is in the support of divisor $2(P_1)-2(\mathcal{O})$,

[1] This is assumed in [22, §3.2]. The proof can be found in Lemma 3.

which is in contradiction with the assumption $x_1 \neq x_4$. [Script 5] in the Appendix verifies the correctness of this formula.

- Case $u_2(x_1) \neq 0$: In this case, we have disjoint divisors, and the generic addition formula in [22, §3.1] applies directly. We have

$$\begin{aligned} q_3 &= x_1 - q_2 - A^2, \\ r_3 &= a_3 + q_2^2 - r_2 - A(B + s_2) + x_1 q_3, \\ s_3 &= A q_3 - B, \\ t_3 &= A r_3 - C \end{aligned} \tag{6}$$

where

$$\begin{aligned} A &= \frac{y_1 - s_2 x_1 - t_2}{x_1^2 + q_2 x_1 + r_2}, \\ B &= s_2 + q_2 A, \\ C &= t_2 + r_2 A. \end{aligned}$$

Note that $x_1^2 + q_2 x_1 + r_2 = u_2(x_1) \neq 0$. [Script 6] in the Appendix verifies the correctness of this formula.

Finally, we remark that the addition of a non-degenerate divisor with a degenerate divisor cannot give the zero divisor by Lemma 2.

3.4 Non-degenerate/non-degenerate inputs

In this section, we examine the last and most common addition, where both inputs are non-degenerate. Let $D_1 = (P_1) + (P_2) - 2(\mathcal{O}) = [u_1(x), v_1(x)] = [x^2 + q_1 x + r_1, s_1 x + t_1]$, and $D_2 = (P_3) + (P_4) - 2(\mathcal{O}) = [u_2(x), v_2(x)] = [x^2 + q_2 x + r_2, s_2 x + t_2]$.

- Case $u_1 = u_2$: This implies that $\{x_1, x_2\} = \{x_3, x_4\}$, though we do not know x_i and y_i explicitly. We continue by investigating v_1 and v_2.
  - Case $v_1 = -v_2$: We have either $v_1(x_1) = -v_2(x_3)$ and $v_1(x_2) = -v_2(x_4)$ or $v_1(x_1) = -v_2(x_4)$ and $v_1(x_2) = -v_2(x_3)$ which implies that $\{y_1, y_2\} = \{-y_3, -y_4\}$. Thus, $D_1 = -D_2$ and $D_3 = 0$.
  - Case $v_1 = v_2$: Similarly, we have either $v_1(x_1) = v_2(x_3)$ and $v_1(x_2) = v_2(x_4)$ or $v_1(x_1) = v_2(x_4)$ and $v_1(x_2) = v_2(x_3)$ which implies that $\{y_1, y_2\} = \{y_3, y_4\}$. Also, note that $y_1 \neq 0$ or $y_2 \neq 0$ because otherwise $\{y_1, y_2\} = \{-y_3, -y_4\}$, which is covered in "Case $v_1 = -v_2$". Thus, $D_1 = D_2$ and the doubling operation $D_3 = 2D_1$ is present. We define the common subexpressions A, B, and C for doubling as in [21, §5].

$$\begin{aligned} A &= \left((q_1^2 - 4r_1 + a_3)q_1 - a_2 + s_1^2\right)(q_1 s_1 - t_1) \\ &\quad + (3q_1^2 - 2r_1 + a_3) r_1 s_1, \\ B &= 2(q_1 s_1 - t_1) t_1 - 2 r_1 s_1^2, \\ C &= \left((q_1^2 - 4r_1 + a_3)q_1 - a_2 + s_1^2\right) s_1 \\ &= + (3q_1^2 - 2r_1 + a_3) t_1. \end{aligned} \tag{7}$$

The doubling formula in [21, §5] is not defined for two cases; when the output is degenerate and when the denominator B vanishes. The approach in [10, §3.1,§3.4] properly handles each case leaving a minor use of polynomial arithmetic. Our little contribution here is to eliminate the polynomial arithmetic and carry out the computation solely with field arithmetic. Therefore, we need to pinpoint algebraically the case where the output is degenerate or non-degenerate. These cases are given as follows.

* Case $B = 0$: B is equal to $-2y_1 y_2$ which corresponds to the resultant$(h + 2v_1, u_1)$ calculated in [10, §3.1]. Now, $B = 0$ induces at least one of the points in the support being a special point which gives the identity when doubled. We assume $y_2 = 0$ without loss of generality and compute x_2 by explicitly calculating $\gcd(u_1, v_1) = x - x_1$ as in [10, §3.1]. We have $x_2 = -t_1/s_1$, $x_1 = -q_1 - x_2$ and $y_1 = v_1(x_1)$. Finally, $D_3 = 2(P_1) - 2(\mathcal{O})$ can be computed with the composition formula (3).
* Case $B \neq 0$: Here, the points in the support are not special points, but a degenerate output is possible. The line "$s = k/(h + 2v) \bmod u$" of Section 3.4 of [10] is subject to accidental cancellation, which may lead to a degenerate output that is again handled implicitly. An explicit approach is to detect the degenerate output beforehand and to generate new formulas free of polynomial arithmetic. For this purpose, we investigate the common subexpression C.
* Case $C = 0$: As stated in [22, §3.3], $C = 0$ implies a degenerate output since the coefficient of the degree 3 term in the intersection parabola $y = dx^3 + ax^2 + bx + c$ is equal to $-C/B$ [22]. The corrected version of the formula in [22, §3.3] is as follows[2]

$$\begin{aligned} x_5 &= 2q_1 + \frac{A^2}{B^2}, \\ y_5 &= x_5(q_1 + x_5)\frac{A}{B} - x_5 s_1 + r_1 \frac{A}{B} - t_1. \end{aligned} \tag{8}$$

[2] Incorrect x_3 is replaced with the correct x_5.

[Script 7] in the Appendix verifies the correctness of this formula.

* Case $C \neq 0$: In this case, the doubling formula in [21, §5] applies for the non-degenerate output D_3 [10, 20].

$$\begin{aligned} q_3 &= 2\frac{A}{C} - \frac{B^2}{C^2}, \\ r_3 &= \frac{A^2}{C^2} + 2q_1\frac{B^2}{C^2} - 2s_1\frac{B}{C}, \\ s_3 &= (r_1 - r_3)\frac{C}{B} - q_3(q_1 - q_3)\frac{C}{B} \\ &\quad + (q_1 - q_3)\frac{A}{B} - s_1, \\ t_3 &= (r_1 - r_3)\frac{A}{B} - r_3(q_1 - q_3)\frac{C}{B} - t_1. \end{aligned} \tag{9}$$

[Script 8] in the Appendix verifies the correctness of this formula.

– Case $v_1 \neq \pm v_2$: Since $\{x_1, x_2\} = \{x_3, x_4\}$ and $P_i = (x_i, y_i)$ are points on the curve, we have $\{y_1, y_2\} = \{\pm y_3, \pm y_4\}$. This gives us four configurations
(i) $\{y_1, y_2\} = \{y_3, y_4\}$,
(ii) $\{y_1, y_2\} = \{-y_3, y_4\}$,
(iii) $\{y_1, y_2\} = \{y_3, -y_4\}$, and
(iv) $\{y_1, y_2\} = \{-y_3, -y_4\}$.
Notice that (i) implies $v_1 = v_2$ and (iv) implies $v_1 = -v_2$. Thus, either (ii) or (iii) holds in this case but not (i) and (iv). Without loss of generality, we may assume $P_2 = -P_4$ and so $D_3 = 2(P_1) - 2(\mathcal{O})$. We compute $x_1 = (t_1 - t_2)/(s_2 - s_1)$ and $y_1 = v_1(x_1)$ as in [10, §3.1]. The doubling of P_1 can be done with the formula (3).

• Case $u_1 \neq u_2$: Here, the supports of D_1 and D_2 are not identical. Thus, we continue with the addition formulas. We define the common subexpressions A, B, and C as in [21, §5].

$$\begin{aligned} A &= (t_1 - t_2)(q_2(q_1 - q_2) \\ &\quad - (r_1 - r_2)) - r_2(q_1 - q_2)(s_1 - s_2), \\ B &= (r_1 - r_2)(q_2(q_1 - q_2) \\ &\quad - (r_1 - r_2)) - r_2(q_1 - q_2)^2, \\ C &= (q_1 - q_2)(t_1 - t_2) - (r_1 - r_2)(s_1 - s_2). \end{aligned} \tag{10}$$

As before, we try to detect the cases where the addition formula in [21, §5] is not defined. For that purpose, we investigate the common subexpressions B and C. The reason that we concentrate on B is illuminated when we write B in terms of x_1, x_2, x_3, and x_4 which is given as follows,

$$B = -(x_2 - x_4)(x_2 - x_3)(x_1 - x_4)(x_1 - x_3).$$

The value $-B$ corresponds to the resultant (u_1, u_2) in [10, §3.1] and becomes zero when $\{P_1, P_2\} \cap \{\pm P_3, \pm P_4\} \neq \emptyset$.

– Case $B = 0$: The supports of the input divisors are joint. Without loss of generality we may assume $x_1 = x_3$ and calculate $x_1 = -(r_1 - r_2)/(q_1 - q_2)$, $y_1 = v_1(x_1)$, $x_3 = x_1$, $y_3 = v_2(x_3)$, $x_2 = -q_1 - x_1$, $y_2 = v_1(x_2)$, $x_4 = -q_2 - x_3$, and $y_4 = v_2(x_4)$ as in [10, §3.1].
* Case $y_1 = -y_3$: Here, $P_1 = (x_1, y_1)$ and $P_3 = (x_1, -y_1)$ are negatives of each other. So, $D_3 = (P_2) + (P_4) - 2(\mathcal{O})$ can be calculated with (2).
* Case $y_1 = y_3$: In this case, $P_1 = (x_1, y_1)$ can be doubled with (3) and $P_2 = (x_2, y_2)$ and $P_4 = (x_4, y_4)$ can be added one by one using the procedure in Sect. 3.3.

– Case $B \neq 0$: Here the supports of the divisors are disjoint. We continue with the investigation of degenerate output. An accidental cancellation occurs in the addition formulas [10, §3.2] when the output is degenerate, just as in doubling. In [22, §3.3], it is stated that $C = 0$ implies a degenerate output since the coefficient $d = -C/B = 0$ of the intersection parabola $y = dx^3 + ax^2 + bx + c$ annihilates the degree 3 term.
* Case $C = 0$: The corrected version of the formula in [22, §3.3] is as follows[3]

$$\begin{aligned} x_5 &= (q_1 + q_2) + \frac{A^2}{B^2}, \\ y_5 &= x_5(q_1 + x_5)\frac{A}{B} - x_5 s_1 + r_1\frac{A}{B} - t_1. \end{aligned} \tag{11}$$

[Script 9] in the Appendix verifies the correctness of this formula.

* Case $C \neq 0$: The addition formula in [21, §5] applies. Furthermore, this is the most frequent case in most cryptographic applications. We reproduce the formulas here to keep the text self-contained.

$$\begin{aligned} q_3 &= (q_1 - q_2) + 2\frac{A}{C} - \frac{B^2}{C^2}, \\ r_3 &= (q_1 - q_2)\frac{A}{C} + \frac{A^2}{C^2} + (q_1 + q_2)\frac{B^2}{C^2} \\ &\quad - (s_1 + s_2)\frac{B}{C}, \end{aligned}$$

[3] Incorrect x_3 is replaced with the correct x_5.

$$
\begin{aligned}
s_3 &= (r_1 - r_3)\frac{C}{B} - q_3(q_1 - q_3)\frac{C}{B} \\
&\quad + (q_1 - q_3)\frac{A}{B} - s_1, \\
t_3 &= (r_1 - r_3)\frac{A}{B} - r_3(q_1 - q_3)\frac{C}{B} - t_1.
\end{aligned} \tag{12}
$$

[Script 10] in the Appendix verifies the correctness of this formula.

4 Jacobian coordinates

The first inversion-free projective formulas for divisor addition on genus 2 hyperelliptic curves are proposed by Lange in [17] where an affine divisor (q_1, r_1, s_1, t_1) is represented as $(Q_1, R_1, S_1, T_1, Z_1)$ satisfying $(\lambda Q_1 : \lambda R_1 : \lambda S_1 : \lambda T_1 : \lambda Z_1) = (q_1 : r_1 : s_1 : t_1 : 1)$ and $\lambda \neq 0$. Lange improved the operation counts by adopting a new weighting where (q_1, r_1, s_1, t_1) is represented as $(Q_1, R_1, S_1, T_1, Z_1, W_1)$ satisfying $(\lambda^2 Q_1 : \lambda^2 R_1 : \lambda^3 \mu S_1 : \lambda^3 \mu T_1 : \lambda Z_1 : \mu W_1) = (q_1 : r_1 : s_1 : t_1 : 1 : 1)$ and $\lambda, \mu \neq 0$ in [19].

Hisil and Costello introduced Jacobian coordinates for hyperelliptic curves of genus 2 in [21]. Their work transfers the weights of x and y coordinates of H across Mumford coordinates and provides a weighted projective representation of divisors. Let (q_1, r_1, s_1, t_1) represent an affine non-degenerate divisor $[x^2 + q_1 x + r_1, s_1 x + t_1] \in J(H)$. Such divisor (q_1, r_1, s_1, t_1) is represented as $(Q_1, R_1, S_1, T_1, Z_1, W_1)$ satisfying $(\lambda^2 Q_1 : \lambda^4 R_1 : \lambda^3 \mu S_1 : \lambda^5 \mu T_1 : \lambda Z_1 : \mu W_1) = (q_1 : r_1 : s_1 : t_1 : 1 : 1)$ and $\lambda, \mu \neq 0$. We refer to [21] for further details.

In this work, we represent degenerate divisors following the same construction. Let (x_1, y_1) represent an affine degenerate divisor $[x - x_1, y_1] \in J(H)$. By adapting the weights above to x, y coordinates, we represent (x_1, y_1) as (X_1, Y_1, Z_1, W_1) satisfying $(\lambda^2 X_1 : \lambda^5 \mu Y_1 : \lambda Z : \mu W) = (x_1 : y_1 : 1 : 1)$ and $\lambda, \mu \neq 0$.

The inversion-free formulas in weighted Jacobian coordinates are given only for the most common addition and doubling (corresponding to Eqs. (9) and (12) resp.) in [21], and only for the cases that contain degenerate divisors in [22]. Moreover, the degenerate divisors in [22] are considered to be in affine coordinates.

By following the same order as in Sect. 3, we present inversion-free formulas with all divisors in weighted Jacobian coordinates for the sake of a complete inversion-free divisor addition algorithm. We omit the cases that are already given in [21, 22] by referring the reader to the mentioned papers. Finally, a complete and inversion-free divisor addition algorithm is given in https://github.com/ozbayelif/jac-on-jac in Magma [27]. The code is optimized to reduce operation counts by eliminating the common subexpressions in the formulas. Throughout this section, we will represent the output divisor as $D_3 = (P_5) + (P_6) - 2(\mathcal{O}) = (Q_3, R_3, S_3, T_3, Z_3, W_3)$ if non-degenerate and as $D_3 = (P_5) - (\mathcal{O}) = (X_5, Y_5, Z_5, W_5)$ if degenerate.

4.1 Degenerate/degenerate inputs

Let the input divisors be represented as $D_1 = (X_1, Y_1, Z_1, W_1)$ and $D_2 = (X_2, Y_2, Z_2, W_2)$. The inversion-free formulas of the two cases $D_1 = D_2$ and $D_1 \neq D_2$ are given in [22, §4.4]; however, the input divisors are considered to be in affine coordinates.

The formula for the case $D_1 = D_2$ given below corresponds to Eq. (3).

$$
\begin{aligned}
Q_3 &= -2X_1 \\
R_3 &= X_1^2 \\
S_3 &= \left(a_1 Z_1^8 + 2a_2 X_1 Z_1^6 + 3a_3 X_1^2 Z_1^4 + 5X_1^4\right) W_1 \\
T_3 &= \left(2a_0 Z_1^{10} + a_1 X_1 Z_1^8 - a_3 X_1^3 Z_1^4 - 3X_1^5\right) W_1 \\
Z_3 &= Z_1 \\
W_3 &= 2Y_1
\end{aligned} \tag{13}
$$

The inversion-free version of Eq. (2) for $D_1 \neq D_2$ is as follows.

$$
\begin{aligned}
Q_3 &= -X_1 Z_2^2 - X_2 Z_1^2 \\
R_3 &= X_1 Z_2^2 X_2 Z_1^2 \\
S_3 &= Y_1 Z_2^5 W_2 - Y_2 Z_1^5 W_1 \\
T_3 &= \left(X_1 Y_2 Z_1^3 W_1 - X_2 Y_1 Z_2^3 W_2\right) Z_1^2 Z_2^2 \\
Z_3 &= Z_1 Z_2 \\
W_3 &= \left(X_1 Z_2^2 - X_2 Z_1^2\right) W_1 W_2
\end{aligned} \tag{14}
$$

4.2 Degenerate/non-degenerate inputs

Let the input divisors be represented as $D_1 = (P_1) - (\mathcal{O}) = (X_1, Y_1, Z_1, W_1)$ and $D_2 = (P_3) - (P_4) - 2(\mathcal{O}) = (Q_2, R_2, S_2, T_2, Z_2, W_2)$. As investigated in Sect. 3.3, there exist 4 cases that can occur.

The case where $P_1 = -P_3$ gives the degenerate output $D_3 = (P_4) - (\mathcal{O})$ whose coordinates can be recovered, as described in "Case $u_2(x_1) = 0$", with the following formula.

$$
\begin{aligned}
X_3 &= X_1 & X_4 &= -Q_2 Z_1^2 - X_3 Z_2^2 \\
Y_3 &= S_2 Z_1^3 X_3 Z_2^2 + T_2 Z_1^5 & Y_4 &= S_2 Z_1^3 X_4 + T_2 Z_1^5 \\
Z_3 &= Z_1 & Z_4 &= Z_1 Z_2 \\
W_3 &= W_2 Z_2^5 & W_4 &= W_2
\end{aligned} \tag{15}
$$

The inversion-free tripling formula in [22], for $P_1 = P_3 = P_4$, leaves the point P_1 to be tripled in affine coordinates. Here, the fully weighted formula is given, which corresponds to Eq. (4).

$$\begin{aligned}
F' &= \left(a_1 Z_1^8 + 2a_2 X_1 Z_1^6 + 3a_3 X_1^2 Z_1^4 + 5X_1^4\right) W_1 \\
F'' &= 2\left(a_2 Z_1^6 + 3a_3 X_1 Z_1^4 + 10X_1^3\right) \\
K &= 8Y_1^3 \\
A &= \left(2F''Y_1^2 - F'^2\right) W_1 \\
B &= 2F'Y_1^2 - AX_1 \\
C &= Y_1 K - (AX_1 + 2B) X_1 W_1 \\
Z_3 &= KZ_1 \\
Q_3 &= 3X_1 K^2 - A^2 \\
R_3 &= a_3 Z_3^4 - \left(4AB - 3X_1\left(Q_3 - X_1 K^2\right)\right) K^2 \\
S_3 &= AQ_3 W_1 - 2BK^2 W_1 \\
T_3 &= AR_3 W_1 - CK^4 \\
W_3 &= W_1
\end{aligned} \tag{16}$$

The following case is where $P_1 = P_3 \neq P_4$. Let $2D_1 = (Q, R, S, T, Z, W)$ be computed with Eq. (13). Then, $2D_1$ and $(P_4) - (\mathcal{O})$ can be added, as in Eq. (5), with the below formula.

$$\begin{aligned}
K &= \left(QZ^2 X_4 Z_4^2 + X_4^2 Z^4 + RZ_4^4\right) W_4 \\
A &= Y_4 Z^5 W - \left(SZ_4^3 X_4 Z^2 + TZ_4^5\right) W_4 \\
B &= SKZ_4 + AQ \\
C &= TKZ_4 + AR \\
L &= KZ_4 W \\
M &= KZW \\
Q_3 &= X_4 M^2 - QL^2 - A^2 \\
R_3 &= \left(a_3 Z^4 + Q^2 - R\right) L^4 - A\left(B + KSZ_4\right) L^2 \\
&\quad + Q_3 X_4 M^2 \\
S_3 &= AQ_3 - BL^2 \\
T_3 &= AR_3 - CL^4 \\
Z_3 &= KZZ_4 W \\
W_3 &= 1
\end{aligned} \tag{17}$$

The last case corresponds to Eq. (6), where the supports are disjoint. The formula in [22] leaves P_1 in affine coordinates. The fully-weighted formula is given below.

$$\begin{aligned}
K &= \left(Q_2 Z_2^2 X_1 Z_1^2 + X_1^2 Z_2^4 + R_2 Z_1^4\right) W_1 \\
A &= Y_1 Z_2^5 W_2 - \left(S_2 Z_1^3 X_1 Z_2^2 + T_2 Z_1^5\right) W_1 \\
B &= S_2 K Z_1 + A Q_2 \\
C &= T_2 K Z_1 + A R_2 \\
L &= K Z_1 W_2 \\
M &= K Z_2 W_2 \\
Q_3 &= X_1 M^2 - Q_2 L^2 - A^2 \\
R_3 &= \left(a_3 Z_2^4 + Q_2^2 - R_2\right) L^4 \\
&\quad - A\left(B + K S_2 Z_1\right) L^2 + Q_3 X_1 M^2 \\
S_3 &= AQ_3 - BL^2 \\
T_3 &= AR_3 - CL^4 \\
Z_3 &= KZ_1 Z_2 W_2 \\
W_3 &= 1
\end{aligned} \tag{18}$$

4.3 Non-degenerate/non-degenerate inputs

Let the input divisors be $D_1 = (P_1) + (P_2) - 2(\mathcal{O}) = (Q_1, R_1, S_1, T_1, Z_1, W_1)$ and $D_2 = (P_3) + (P_4) - 2(\mathcal{O}) = (Q_2, R_2, S_2, T_2, Z_2, W_2)$.

We omit the cases where the output is degenerate and refer the reader to [22, §4.3]. Likewise, we refer to [21] for the common subexpressions in Jacobian coordinates and the formulas for the common cases. Throughout this section, we use the notation $P_i = (X_{P_i}, Y_{P_i}, Z_{P_i}, W_{P_i})$ for the points in the supports to avoid confusion between the Z, W coordinates of points and divisors.

We continue with doubling where $u_1 = u_2$ and $v_1 = v_2$. The case $B = 0$ does not involve a degenerate divisor, yet it is a rare case where the common addition formulas do not work. Hence, the case is missing in the literature. When $B = 0$, the result $D_3 = 2(P_1) - 2(\mathcal{O})$ can be calculated by making the coordinates of P_1 explicit as below and doubling it with Eq. (13).

$$\begin{aligned}
X_{P_2} &= -T_1 \\
X_{P_1} &= -\left(Q_1 S_1 - T_1\right) S_1 \\
Y_{P_1} &= \left(S_1 T_1 + X_{P_1}\right) S_1^4 \\
Z_{P_1} &= S_1 Z_1 \\
W_{P_1} &= W_1
\end{aligned} \tag{19}$$

The addition in the case $u_1 = u_2$, $v_1 \neq \pm v_2$ gives $D_3 = 2(P_1) - 2(\mathcal{O})$. The coordinates of P_1 can be computed with

the following formula and can be doubled with Eq. (13).

$$\begin{aligned} K &= S_1 Z_2^3 W_2 - S_2 Z_1^3 W_1 \\ X_{P_1} &= \left(T_2 Z_1^5 W_1 - T_1 Z_2^5 W_2\right) K \\ Y_{P_1} &= \left(S_1 X_{P_1} + T_1 Z_2^2 K^2\right) K^3 Z_2^3 \\ Z_{P_1} &= K Z_1 Z_2 \\ W_{P_1} &= W_1 \end{aligned} \tag{20}$$

Now we investigate addition where $u_1 \neq u_2$. We focus on the case $B = 0$ which indicates that the input divisors share exactly one common point in their supports. The coordinates of the points in the supports can be calculated as below.

$$\begin{aligned} K &= Q_1 Z_2^2 - Q_2 Z_1^2 \\ X_{P_1} &= \left(R_2 Z_1^4 - R_1 Z_2^4\right) K \\ Y_{P_1} &= \left(S_1 X_{P_1} + T_1 K^2 Z_2^2\right) K^3 Z_2^3 \\ Z_{P_1} &= Z_1 Z_2 K \\ W_{P_1} &= W_1 \\ X_{P_2} &= X_{P_1} \\ Y_{P_2} &= \left(S_2 X_{P_2} + T_2 K^2 Z_1^2\right) K^3 Z_1^3 \\ Z_{P_2} &= Z_{P_1} \\ W_{P_2} &= W_2 \\ X_{P_3} &= -Q_1 K^2 Z_2^2 - X_{P_1} \\ Y_{P_3} &= \left(S_1 X_{P_3} + T_1 K^2 Z_2^2\right) K^3 Z_2^3 \\ Z_{P_3} &= Z_{P_1} \\ W_{P_3} &= W_1 \\ X_{P_4} &= -Q_2 K^2 Z_1^2 - X_{P_2} \\ Y_{P_4} &= \left(S_2 X_{P_4} + T_2 K^2 Z_1^2\right) K^3 Z_1^3 \\ Z_{P_4} &= Z_{P_1} \\ W_{P_4} &= W_2 \end{aligned} \tag{21}$$

In the case where $Y_{P_1} = -Y_{P_3}$, the result is $D_3 = (P_2) + (P_4) - 2(\mathcal{O})$ and can be calculated with Eq. (14).

If $Y_{P_1} = Y_{P_3}$, the result can be calculated by doubling P_1 with Eq. (13) and adding P_2 and P_3 with Eq. (14).

5 Cost analysis

The divisor addition on genus 2 hyperelliptic curves is intricate, with plenty of different cases. Although rare cases have a low chance of occurrence, handling them in cryptographic applications is vital. While some of the rare cases are quite costly to handle, some others are not.

In Fig. 1; we illustrate all the cases of divisor addition on genus 2 hyperelliptic curves; we present the operation counts and point out the gaps in the literature, which are filled with this work. The dashed edges represent the missing cases in the literature that are proposed with fully-weighted formulas in Jacobian coordinates here. We refer to; degenerate/degenerate addition as $1+1$, degenerate/non-degenerate addition as $1+2$, and non-degenerate/non-degenerate addition as $2+2$. The letter '**M**' stands for the number of field multiplications; '**S**' for field squarings; and '**D**' for addition or multiplication with constants, respectively. Each edge represents a condition to be checked before determining the present case, with respect to the algorithm in Sect. 3. The operation counts given below the edges denote the cost of checking the condition. In some cases, the operations required for determining the case to be handled also provide common subexpressions, which are shared with the addition formulas in that context. Each leaf of the tree corresponds to a case in Sect. 3. The operation counts of each case are given in the leaves.

The case where $u_1 \neq u_2$, $B = 0$, and $P_1 = P_3$ leads to 8 different cases enumerated with '*' in front. In that case, $D_1 = (P_1) + (P_2) - 2(\mathcal{O})$ and $D_2 = (P_1) + (P_4) - 2(\mathcal{O})$. Adding D_1 and D_2 requires doubling P_1 and adding P_2 and P_4 to the result. Here, 2 cases may occur while the addition of P_2: either $P_1 = P_2$ or $P_1 \neq P_2$ ($P_1 = -P_2$ is not possible since D_1 would not be reduced otherwise). When $P_1 = P_2$, the addition of $2(P_1) - 2(\mathcal{O})$ and $(P_2) - (\mathcal{O})$ corresponds to the tripling in Eq. (4). If $P_1 \neq P_2$, the addition can be done with Eq. (6). Let $(P_5) + (P_6) - 2(\mathcal{O})$ be the result for both cases. The result may or may not contain a point in its support joint with $(P_4) - (\mathcal{O})$. That leads to 4 cases for both:

- *1 and *5: $-P_4 = P_5$ (output P_6),
- $P_4 = P_5$
  - *2 and *6: $P_5 = P_6$ (Eq. (4)),
  - *3 and *7: $P_5 \neq P_6$ (Eq. (5)),
- *4 and *8: $\pm P_4 \neq P_5$ (Eq. (6)).

We note that P_5 and P_6 can be interchanged without loss of generality.

The operations needed for the detection of the cases are also taken into account in contrast to the previous works, which present the operation counts for specific cases. This may lead to differences with the operation counts given in the literature. However, exceptional outputs may be produced when assumptions on the inputs are made without checking if the conditions are met. Thus, it is sensible to include the cost of case detection to demonstrate the performance of our complete algorithm.

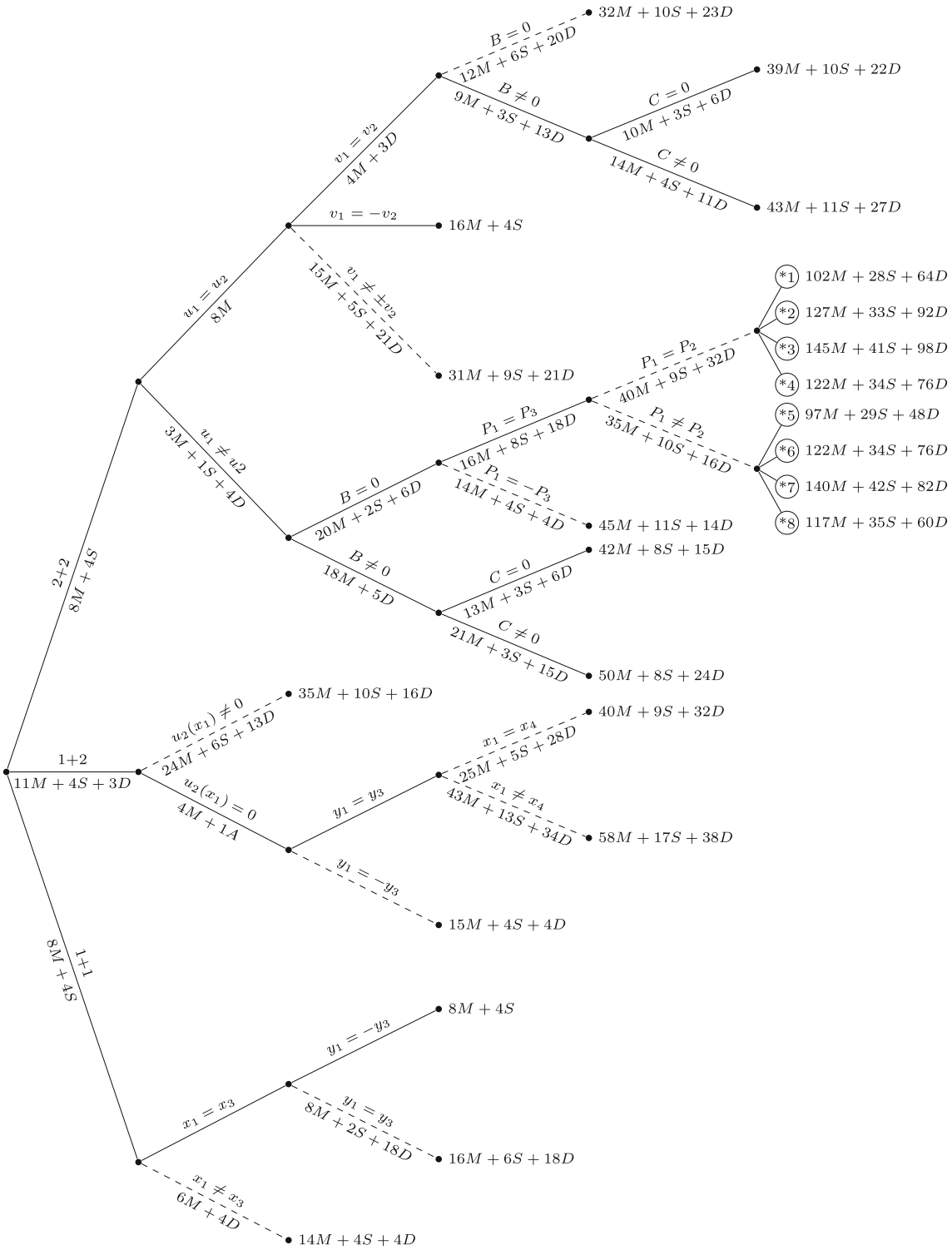

Fig. 1 Divisor addition cases

6 Conclusion

This paper introduced explicit and inversion-free formulas for the complete group law of genus 2 hyperelliptic curves. The formulas are derived from Cantor's algorithm and presented case by case in regard to whether the divisors are degenerate or non-degenerate. This work can be seen as a completion of the works started in [21] and [22] on Jacobian coordinates.

Any cryptographic application which enforces using a prime order genus 2 Jacobian is amenable to introducing exceptional situations, such as the addition of two divisors with joint support. Our formulas provide an efficient fallback that eliminates the need for polynomial arithmetic required by Cantor's algorithm. Although the formulas we present consist of rare cases, a cryptographic application based on an operation such as digital signature verification is expected to handle all cases properly. In such a scenario, our complete algorithm can be used to prevent active attacks based on faulty inputs which target to trigger an exceptional output.

Acknowledgements This work grew upon a question raised by Tanja Lange to the second author in ASIACRYPT 2014. We also thank the reviewers of this work for their suggestions and corrections. Part of this study was done while the second author was at Yaşar University.

Funding Open Access funding enabled and organized by Projekt DEAL.

Appendix: Verification Scripts

This section contains Magma [27] scripts that verify the formulas presented in the text and assist in proving selected theorems. These scripts can be executed freely online through Web-Magma (http://magma.maths.usyd.edu.au/calc/), noting that each script requires [Script 1] to be loaded in advance.

Script 1 Cantor's Algorithm for genus 2

```
Cantor:=function(u1,v1,u2,v2,f)
    d1,e1,e2:=Xgcd(u1,u2); d,c1,c2:=Xgcd(d1,v1+v2);
    s1:=c1*e1; s2:=c1*e2; s3:=c2;
    u:=(u1*u2) div d^2; v:=((s1*u1*v2+s2*u2*v1+s3*(v1*v2+f)) div d) mod u;
    while Degree(u) gt 2 do
        u:=(f-v^2) div u; v:=(-v) mod u;
    end while;
    u:=Normalize(u);
    if Degree(u) eq 0 then
        return 1,0;
    elif Degree(u) eq 1 then
        return -Coefficient(u,0),v;
    elif Degree(u) eq 2 then
        return Coefficient(u,1),Coefficient(u,0),
            Coefficient(v,1),Coefficient(v,0);
    else
        print "error: Non-reduced divisor!";
    end if;
end function;
```

Script 2 Verification of (3) and Sect. 3.2 "Case $y_1 = y_2$"

```
F<a3,a2,a1,a0,x1,y1>:=FunctionField(Rationals(),6);
_<x>:=PolynomialRing(F);
q,r,s,t:=Cantor(x-x1,y1,x-x1,y1,x^5+a3*x^3+a2*x^2+a1*x+a0);
/*** START OF FORMULA ***/
q3:=-2*x1;
r3:=x1^2;
s3:=(5*x1^4+3*a3*x1^2+2*a2*x1+a1)/(2*y1);
t3:=(-3*x1^5-a3*x1^3+a1*x1+2*a0)/(2*y1);
/*** END OF FORMULA ***/
Q:=RingOfFractions(quo<Parent(Numerator(q))|
   y1^2-(x1^5+a3*x1^3+a2*x1^2+a1*x1+a0)>);
Q!(q-q3) eq 0; Q!(r-r3) eq 0; Q!(s-s3) eq 0; Q!(t-t3) eq 0;
```

Script 3 Verification of (2) and Sect. 3.2 "Case $x_1 \neq x_2$"

```
F<a3,a2,a1,a0,x1,y1,x2,y2>:=FunctionField(Rationals(),8);
_<x>:=PolynomialRing(F);
q,r,s,t:=Cantor(x-x1,y1,x-x2,y2,x^5+a3*x^3+a2*x^2+a1*x+a0);
/*** START OF FORMULA ***/
q3:=-x1-x2;
r3:=x1*x2;
s3:=(y1-y2)/(x1-x2);
t3:=(x1*y2-x2*y1)/(x1-x2);
/*** END OF FORMULA ***/
Q:=RingOfFractions(quo<Parent(Numerator(q))|
   y1^2-(x1^5+a3*x1^3+a2*x1^2+a1*x1+a0),
   y2^2-(x2^5+a3*x2^3+a2*x2^2+a1*x2+a0)>);
Q!(q-q3) eq 0; Q!(r-r3) eq 0; Q!(s-s3) eq 0; Q!(t-t3) eq 0;
```

Script 4 Verification of (4)

```
F<a3,a2,a1,a0,x1,y1>:=FunctionField(Rationals(),6);
_<x>:=PolynomialRing(F);
q1:=-2*x1; r1:=x1^2;
s1:=(5*x1^4+3*a3*x1^2+2*a2*x1+a1)/(2*y1);
t1:=(-3*x1^5-a3*x1^3+a1*x1+2*a0)/(2*y1);
q,r,s,t:=Cantor(x^2+q1*x+r1,s1*x+t1,x-x1,y1,x^5+a3*x^3+a2*x^2+a1*x+a0);
/*** START OF FORMULA ***/
fdashx1:=5*x1^4+3*a3*x1^2+2*a2*x1+a1;
fddashx1:=20*x1^3+6*a3*x1+2*a2;
A:=(2*y1^2*fddashx1-fdashx1^2)/(8*y1^3);
B:=fdashx1/(2*y1)-2*A*x1;
C:=y1-A*x1^2-B*x1;
q3:=3*x1-A^2;
r3:=a3-2*A*B+3*x1*(q3-x1);
s3:=A*q3-B;
t3:=A*r3-C;
/*** END OF FORMULA ***/
Q:=RingOfFractions(quo<Parent(Numerator(q))|
   y1^2-(x1^5+a3*x1^3+a2*x1^2+a1*x1+a0)>);
Q!(q-q3) eq 0; Q!(r-r3) eq 0; Q!(s-s3) eq 0; Q!(t-t3) eq 0;
```

Script 5 Verification of (5)

```
P<a3,a2,a1,a0,q1,r1,s1,t1,x3,y3>:=PolynomialRing(Rationals(),10);
Q<a3,a2,a1,a0,q1,r1,s1,t1,x3,y3>:=RingOfFractions(quo<P|
    x3^2+q1*x3+r1, y3-(s1*x3+t1),    /* x1=x3 and y1=y3 */
    r1*(s1^2+q1^3-(2*r1-a3)*q1-a2)-(t1^2-a0),
    q1*(s1^2+q1^3-(3*r1-a3)*q1-a2)-(2*s1*t1-r1*(r1-a3)-a1)>);
_<x>:=PolynomialRing(Q);
q,r,s,t:=Cantor(x^2+q1*x+r1,s1*x+t1,x-x3,y3,x^5+a3*x^3+a2*x^2+a1*x+a0);
/*** START OF FORMULA ***/
x1:=x3; y1:=y3; x2:=-q1-x1; y2:=s1*x2+t1;
q2:=-2*x1;
r2:=x1^2;
s2:=(5*x1^4+3*a3*x1^2+2*a2*x1+a1)/(2*y1);
t2:=(-3*x1^5-a3*x1^3+a1*x1+2*a0)/(2*y1);
A:=(y2-(s2*x2+t2))/(x2^2+q2*x2+r2);
B:=s2+q2*A;
C:=t2+r2*A;
q3:=x2-q2-A^2;
r3:=a3+q2^2-r2-A*(B+s2)+x2*q3;
s3:=A*q3-B;
t3:=A*r3-C;
/*** END OF FORMULA ***/
Q!(q-q3) eq 0; Q!(r-r3) eq 0; Q!(s-s3) eq 0; Q!(t-t3) eq 0;
```

Script 6 Verification of (6)

```
F<a3,a2,a1,a0,x1,y1,x2,y2,x3,y3>:=FunctionField(Rationals(),10);
_<x>:=PolynomialRing(F);
q1:=-x1-x2; r1:=x1*x2; s1:=(y1-y2)/(x1-x2); t1:=(x1*y2-y1*x2)/(x1-x2);
q,r,s,t:=Cantor(x^2+q1*x+r1,s1*x+t1,x-x3,y3,x^5+a3*x^3+a2*x^2+a1*x+a0);
/*** START OF FORMULA ***/
A:=(y3-(s1*x3+t1))/(x3^2+q1*x3+r1);
B:=s1+q1*A;
C:=t1+r1*A;
q3:=x3-q1-A^2;
r3:=a3+q1^2-r1-A*(B+s1)+x3*q3;
s3:=A*q3-B;
t3:=A*r3-C;
/*** END OF FORMULA ***/
F!(q-q3) eq 0; F!(r-r3) eq 0; F!(s-s3) eq 0; F!(t-t3) eq 0;
```

Script 7 Verification of (8)

```
P<a3,a2,a1,a0,q1,r1,s1,t1>:=PolynomialRing(Rationals(),8);
Q<a3,a2,a1,a0,q1,r1,s1,t1>:=RingOfFractions(quo<P|
     ((q1^2-4*r1+a3)*q1-a2+s1^2)*s1+(3*q1^2-2*r1+a3)*t1, /* C=0 */
     r1*(s1^2+q1^3-(2*r1-a3)*q1-a2)-(t1^2-a0),
     q1*(s1^2+q1^3-(3*r1-a3)*q1-a2)-(2*s1*t1-r1*(r1-a3)-a1)>);
_<x>:=PolynomialRing(Q);
x,y:=Cantor(x^2+q1*x+r1,s1*x+t1,x^2+q1*x+r1,
            s1*x+t1,x^5+a3*x^3+a2*x^2+a1*x+a0);
/*** START OF FORMULA ***/
A:=((q1^2-4*r1+a3)*q1-a2+s1^2)*(q1*s1-t1)+(3*q1^2-2*r1+a3)*r1*s1;
B:=2*(q1*s1-t1)*t1-2*r1*s1^2;
x5:=2*q1+A^2/B^2;
y5:=(A/B*(q1+x5)-s1)*x5+(A/B*r1-t1);
/*** END OF FORMULA ***/
Q!(x-x5) eq 0; Q!(y-y5) eq 0;
```

Script 8 Verification of (9)

```
P<a3,a2,a1,a0,q1,r1,s1,t1>:=PolynomialRing(Rationals(),8);
Q<a3,a2,a1,a0,q1,r1,s1,t1>:=RingOfFractions(quo<P|
     r1*(s1^2+q1^3-(2*r1-a3)*q1-a2)-(t1^2-a0),
     q1*(s1^2+q1^3-(3*r1-a3)*q1-a2)-(2*s1*t1-r1*(r1-a3)-a1)>);
_<x>:=PolynomialRing(Q);
q,r,s,t:=Cantor(x^2+q1*x+r1,s1*x+t1,x^2+q1*x+r1,
                s1*x+t1,x^5+a3*x^3+a2*x^2+a1*x+a0);
/*** START OF FORMULA ***/
A:=((q1^2-4*r1+a3)*q1-a2+s1^2)*(q1*s1-t1)+(3*q1^2-2*r1+a3)*r1*s1;
B:=2*(q1*s1-t1)*t1-2*r1*s1^2;
C:=((q1^2-4*r1+a3)*q1-a2+s1^2)*s1+(3*q1^2-2*r1+a3)*t1;
q3:=2*(A/C)-B^2/C^2;
r3:=A^2/C^2+2*q1*B^2/C^2-2*s1*(B/C);
s3:=(r1-r3)*(C/B)-q3*(q1-q3)*(C/B)+(q1-q3)*(A/B)-s1;
t3:=(r1-r3)*(A/B)-r3*(q1-q3)*(C/B)-t1;
/*** END OF FORMULA ***/
Q!(q-q3) eq 0; Q!(r-r3) eq 0; Q!(s-s3) eq 0; Q!(t-t3) eq 0;
```

Script 9 Verification of (11)

```
P<a3,a2,a1,a0,t1,t2,q1,r1,s1,q2,r2,s2>:=PolynomialRing(Rationals(),12);
Q<a3,a2,a1,a0,t1,t2,q1,r1,s1,q2,r2,s2>:=RingOfFractions(
    quo<P|(q1-q2)*(t1-t2)-(r1-r2)*(s1-s2)>); /* C=0 */
_<x>:=PolynomialRing(Q);
x,y:=Cantor(x^2+q1*x+r1,s1*x+t1,
            x^2+q2*x+r2,s2*x+t2,x^5+a3*x^3+a2*x^2+a1*x+a0);
/*** START OF FORMULA ***/
A:=(t1-t2)*(q2*(q1-q2)-(r1-r2))-r2*(q1-q2)*(s1-s2);
B:=(r1-r2)*(q2*(q1-q2)-(r1-r2))-r2*(q1-q2)^2;
x5:=(q1+q2)+A^2/B^2;
y5:=(A/B*(q1+x5)-s1)*x5+(A/B*r1-t1);
/*** END OF FORMULA ***/
Q!(x-x5) eq 0; Q!(y-y5) eq 0;
```

Script 10 Verification of (12)

```
F<a3,a2,a1,a0,q1,r1,s1,t1,q2,r2,s2,t2>:=FunctionField(Rationals(),12);
_<x>:=PolynomialRing(F);
q,r,s,t:=Cantor(x^2+q1*x+r1,s1*x+t1,x^2+q2*x+r2,
                s2*x+t2,x^5+a3*x^3+a2*x^2+a1*x+a0);
/*** START OF FORMULA ***/
A:=(t1-t2)*(q2*(q1-q2)-(r1-r2))-r2*(q1-q2)*(s1-s2);
B:=(r1-r2)*(q2*(q1-q2)-(r1-r2))-r2*(q1-q2)^2;
C:=(q1-q2)*(t1-t2)-(r1-r2)*(s1-s2);
q3:=(q1-q2)+2*(A/C)-B^2/C^2;
r3:=(q1-q2)*(A/C)+A^2/C^2+(q1+q2)*B^2/C^2-(s1+s2)*(B/C);
s3:=(r1-r3)*(C/B)-q3*(q1-q3)*(C/B)+(q1-q3)*(A/B)-s1;
t3:=(r1-r3)*(A/B)-r3*(q1-q3)*(C/B)-t1;
/*** END OF FORMULA ***/
F!(q-q3) eq 0; F!(r-r3) eq 0; F!(s-s3) eq 0; F!(t-t3) eq 0;
```

References

1. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) Advances in Cryptology—CRYPTO '85 Proceedings, pp. 417–426. Springer, Berlin (1986). https://doi.org/10.1007/3-540-39799-X_31
2. Koblitz, N.: Elliptic curve cryptosystems. Math. Comput. **48**(177), 203–209 (1987). https://doi.org/10.2307/2007884
3. Koblitz, N.: Hyperelliptic cryptosystems. J. Cryptol. **1**(3), 139–150 (1989). https://doi.org/10.1007/BF02252872
4. Gaudry, P.: An algorithm for solving the discrete log problem on hyperelliptic curves. In: Preneel, B. (ed.) Advances in Cryptology—EUROCRYPT 2000, pp. 19–34. Springer, Berlin (2000). https://doi.org/10.1007/3-540-45539-6_2
5. Cantor, D.G.: Computing in the Jacobian of a hyperelliptic curve. Math. Comput. **48**(177), 95–101 (1987). https://doi.org/10.2307/2007876
6. Mumford, D.: Tata lectures on theta II. In: Progress in Mathematics, vol. 43. Birkhauser, Boston (1984). https://doi.org/10.1007/978-0-8176-4578-6
7. Paulus, S., Stein, A.: Comparing real and imaginary arithmetics for divisor class groups of hyperelliptic curves. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 576–591. Springer, Berlin (1998). https://doi.org/10.1007/BFb0054894
8. Smart, N.: On the performance of hyperelliptic cryptosystems. In: Stern, J. (ed.) Advances in Cryptology—EUROCRYPT '99. LNCS, vol. 1592, pp. 165–175. Springer, Berlin (1999). https://doi.org/10.1007/3-540-48910-X_12
9. Nagao, K.: Improving group law algorithms for Jacobians of hyperelliptic curves. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 439–448. Springer, Berlin (2000). https://doi.org/10.1007/10722028_28
10. Lange, T.: Efficient arithmetic on hyperelliptic curves. Cryptology ePrint archive, report 2002/107. https://ia.cr/2002/107 (2002)
11. Spallek, A.: Kurven vom geschlecht 2 und ihre anwendung in public-key-kryptosystemen. Ph.D. thesis, University of Duisburg-Essen, Germany (1994). https://d-nb.info/943571049
12. Gaudry, P., Harley, R.: Counting points on hyperelliptic curves over finite fields. In: Bosma, W. (ed.) Algorithmic Number Theory, pp. 313–332. Springer, Berlin (2000)
13. Lange, T.: Efficient arithmetic on genus 2 hyperelliptic curves over finite fields via explicit formulae. Cryptology ePrint archive, paper 2002/121 (2002). https://eprint.iacr.org/2002/121
14. Miyamoto, Y.: A fast addition algorithm of genus two hyperelliptic curves. In: Proceedings of SCIS2002 (2002)
15. Matsuo, K., Chao, J., Tsujii, S.: Fast genus two hyperelliptic curve cryptosystems, vol. 2001(75), pp. 89–96 (2001)
16. Takahashi, M.: Improving Harley algorithms for Jacobians of genus 2 hyperelliptic curves. In: Proceedings of SCIS2002, pp. 155–160. IEICE Japan (2002)
17. Lange, T.: Inversion-free arithmetic on genus 2 hyperelliptic curves. Cryptology ePrint archive, report 2002/147 (2002). https://eprint.iacr.org/2002/147
18. Wollinger, T.J.: Software and hardware implementation of hyperelliptic curve cryptosystems. Ph.D. thesis, Ruhr University Bochum (2004)
19. Lange, T.: Weighted coordinates on genus 2 hyperelliptic curves. Cryptology ePrint archive, report 2002/153 (2002). https://eprint.iacr.org/2002/153
20. Costello, C., Lauter, K.: Group law computations on Jacobians of hyperelliptic curves. In: Miri, A., Vaudenay, S. (eds.) Selected Areas in Cryptography, pp. 92–117. Springer, Berlin (2012). https://doi.org/10.1007/978-3-642-28496-0_6
21. Hisil, H., Costello, C.: Jacobian coordinates on genus 2 curves. J. Cryptol. **30**, 572–600 (2017). https://doi.org/10.1007/s00145-016-9227-7
22. Hu, Z., Lin, D., Zhao, C.-A.: Fast scalar multiplication of degenerate divisors for hyperelliptic curve cryptosystems. Appl. Math. Comput. **404**, 126239 (2021). https://doi.org/10.1016/j.amc.2021.126239
23. Gaudry, P.: Fast genus 2 arithmetic based on theta functions. J. Math. Cryptol. **1**(3), 243–265 (2007). https://doi.org/10.1515/JMC.2007.012
24. Bernstein, D.J., Chuengsatiansup, C., Lange, T., Schwabe, P.: Kummer strikes back: new DH speed records. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology—ASIACRYPT 2014, pp. 317–337. Springer, Berlin (2014). https://doi.org/10.1007/978-3-662-45611-8_17
25. Menezes, A., Zuccherato, R., Wu, Y.-H.: An elementary introduction to hyperelliptic curves. In: Koblitz, N. (ed.) Algebraic Aspects of Cryptography, pp. 155–178. Springer, Berlin (1998)
26. Koblitz, N., Menezes, A.J., Wu, Y.-H., Zuccherato, R.J.: Algebraic Aspects of Cryptography. Springer, Berlin (1998). https://doi.org/10.1007/978-3-662-03642-6
27. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. J. Symb. Comput. **24**(3–4), 235–265 (1997). https://doi.org/10.1006/jsco.1996.0125. (**Computational algebra and number theory (London, 1993)**)