YAŞAR UNIVERSITY

GRADUATE SCHOOL

MASTER THESIS

COMPLETE GROUP LAW FOR GENUS 2 JACOBIANS ON JACOBIAN COORDINATES

ELİF ÖZBAY GÜRLER

THESIS ADVISOR: ASSIST. PROF. (PHD) HÜSEYİN HIŞIL

COMPUTER ENGINEERING

PRESENTATION DATE: 05.06.2023

BORNOVA / İZMİR

JUNE 2023

We certify that, as the jury, we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Jury Members:	**Signature:**
Assist. Prof. Hüseyin HIŞIL, Ph.D. Yaşar University	. . . . .
Prof. Ahmet Hasan KOLTUKSUZ, Ph.D. Yaşar University	. . . . .
Assist. Prof. Işıl ÖZ , Ph.D. İzmir Institute of Technology	. . . . .

Prof. Yücel Öztürkoğlu, Ph.D.

Director of the Graduate School

ABSTRACT

Complete Group Law for Genus 2 Jacobians on Jacobian Coordinates

Özbay Gürler, Elif
MSc, Computer Engineering
Advisor: Assist. Prof. (PhD) Hüseyin HIŞIL
June 2023

Hyperelliptic curves of genus g are algebraic curves with $g \geq 1$, defined by the equation $y^2 = f(x)$, where $f(x)$ is a polynomial of degree $2g + 1$. Hyperelliptic curves have received much attention due to their rich structure and applications in cryptography and coding theory. Genus 2 hyperelliptic curves are particularly appealing because they offer a good balance between security and efficiency in cryptographic protocols. On the other hand, the Kummer surface remains the speed leader in curve-based cryptosystems, indicating the unexplored potential of genus 2 hyperelliptic curves. While the group law in affine coordinates on hyperelliptic curves provides a general idea of the performance of such curves, inversion-free formulas in projective coordinates are mainly used in cryptographic applications. The Jacobian coordinates have shown a significant reduction in operation counts of divisor addition on genus 2 curves; however, the coordinates are only applied to the common case and the cases that include a degenerate divisor, which leaves out a number of rare cases. This thesis presents explicit, complete group law formulas for hyperelliptic curves of genus 2, both in affine and Jacobian coordinates. The formulas do not require using polynomial arithmetic operations such as resultant, mod, or gcd but only explicit field operations. The interpretation of the group law through geometry, polynomial arithmetic, and divisor arithmetic is also presented here, which provides an insight that offers a deeper understanding of the structure of the curve. This work provides a framework that can be put directly into practice for the efficient implementation of cryptographic protocols.

Keywords: Group law, genus 2, hyperelliptic curves, explicit formulas, complete formulas, inversion-free, Jacobian coordinates

ÖZ

Genus 2 Jacobiyenler için Jacobiyen Koordinatlarda Tam Grup Kanunu

Özbay Gürler, Elif
Yüksek Lisans, Bilgisayar Mühendisliği
Danışman: Dr. Öğretim Üyesi Dr. Hüseyin HIŞIL
Haziran 2023

Genus g hipereliptik eğriler, $g \geq 1$ olmak üzere, $y^2 = f(x)$ denklemiyle tanımlanan cebirsel eğrilerdir. Burada $f(x)$, derecesi $2g+1$ olan bir polinomdur. Hipereliptik eğriler, zengin geometrik yapıları, kriptografi ve kodlama teorisindeki uygulamaları nedeniyle büyük ilgi görmüştür. Özellikle genus 2 hipereliptik eğriler, güvenlik ve verimlilik arasında iyi bir denge sunması sebebiyle öne çıkmaktadır. Öte yandan Kummer yüzeyi, eğri tabanlı kriptosistemlerde hız lideridir ve bu da hipereliptik eğrilerin henüz tam olarak keşfedilmemiş potansiyelinin bir göstergesidir. Hipereliptik eğriler üzerinde afin koordinatlarda grup kanunu, bu tür eğrilerin performansı hakkında genel bir fikir vermektedir. Buna karşın, kriptografik uygulamalarda, ters alma işlemi içermeyen, projektif koordinatlar üzerinde tanımlanan formüller kullanılır. Jacobiyen koordinatlar, her ne kadar bölen toplama için gereken işlem sayısında belirgin bir azalma sağlasa da, yalnızca genel toplama ve dejenere bölen içeren toplama operasyonlarına uygulanmıştır. Bu çerçevede, nadir oluşan bazı özel durumlar göz ardı edilmiştir. Bu tezde, hem afin hem de Jacobiyen koordinatlar cinsinden genus 2 hipereliptik eğriler için açık grup kanunu formülleri ortaya konmuştur. Formüller; resultant, mod ve EBOB gibi polinom aritmetiği hesaplamaları içermemekte olup, salt açık cisim işlemleri içermektedir. Grup kanunu; geometrik, polinom aritmetiği ve bölen aritmetiği yoluyla açıklanmış, ve bu sayede, eğrinin yapısının bütünüyle anlaşılması için bir girişim yapılmıştır. Bu çalışma, kriptografik protokollerin gerçeklenmesi için doğrudan uygulamaya konulabilir bir çerçeve sunmaktadır.

Anahtar Kelimeler: Grup kanunu, genus 2, hipereliptik eğriler, açık formüller, tam formüller, ters alma içermeyen, Jacobiyen koordinatlar

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to Assist. Prof. Hüseyin HIŞIL, who brought me to this point with his guidance and made things that seemed unachievable easy for me. From the day I met him, he introduced me to all sorts of things and taught me how to survive in the vast ocean of research. I can't thank him enough for all he's done for me; he's a great supervisor. I am also thankful to my jury members, Prof. Ahmet Hasan KOLTUKSUZ, and Assist. Prof. Işıl ÖZ for the expertise and insight they generously provided to me.

I also want to thank my beloved husband, Orkun, for being with me through my good and bad times. He encouraged me to do the double major, a turning point in my life, and he did not spare me his support for a second. Without him, I wouldn't be able to discover myself.

I am grateful to my dearest family for always trusting me. My parents, Müge and Ekrem, my brother Hüseyin, and my aunts Hale and Şule did their best for me without hesitation.

Last but not least, thank you to my friends for never making me feel alone and cheering me up even in the toughest times.

Elif Özbay Gürler
İzmir, 2023

TEXT OF OATH

I declare and honestly confirm that my study, titled "Complete Group Law for Genus 2 Jacobians on Jacobian Coordinates" and presented as a Master's Thesis, has been written without applying to any assistance inconsistent with scientific ethics and traditions. I declare, to the best of my knowledge and belief, that all content and ideas drawn directly or indirectly from external sources are indicated in the text and listed in the list of references.

Elif Özbay Gürler

Signature:

_____

June 05, 2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF CODES

SYMBOLS AND ABBREVIATIONS

ABBREVIATIONS:

ECC	Elliptic Curve Cryptography
ECM	Elliptic Curve Factorization Method
DLP	Discrete Logarithm Problem
HECC	Hyperelliptic Curve Cryptosystem
GCD	Greatest Common Divisor
CRT	Chinese Remainder Theorem

SYMBOLS:

g	Genus
$[k]P$	Scalar multiplication of point P with k
K	Field
$\bar{K}$	Algebraic closure of K
$\mathcal{O}$	Point at infinity
$H(\bar{K})$	Point set of H defined on $\bar{K}$
$H(K)$	Point set of H defined on K
0	Zero divisor
(f)	Principal divisor of f
$Div_{\bar{K}}(H)$	Set of all divisors of H
$Div^0_{\bar{K}}(H)$	Set of all degree 0 divisors of H
$Prin_{\bar{K}}(H)$	Set of all principal divisors of H
$Pic_{\bar{K}}(H)$	Picard group of H
$J(H)$	Jacobian of H

CHAPTER 1
INTRODUCTION

As one of the main problems of today, secure communication has been in the interest of many disciplines, especially mathematics and computer sciences. The field of cryptography has been producing cryptosystems based on mathematical foundations for years to ensure the confidentiality of data, and many tasks in our daily lives depend on these cryptosystems. In general, cryptosystems are divided into two: symmetric and asymmetric cryptosystems. Symmetric cryptography generates one key that parties share, while in asymmetric (or public-key) cryptography, both parties have their own secret and public keys.

Public-key cryptography was first introduced by Diffie and Hellman (1976) and is still being used in communication systems. Elliptic curves are offered for the use of public-key cryptography by Miller (1986) and then by Koblitz (1987). Elliptic Curve Cryptography (ECC) depends on the algebraic structure of the point set of an elliptic curve over a finite field. This set of points forms a group, with the group operation being point addition. The scalar multiplication of a point is derived from the addition, and k times a point P is denoted as $[k]P$. The security of an ECC scheme is based on the difficulty of finding k where the point P and $[k]P$ are known. Let $Q = [k]P$. Now, finding k when only P and Q are known, is conjectured to be intractable, and named as the (hyper)elliptic curve discrete logarithm problem (DLP). Accordingly, scalar multiplication has been studied extensively, and its acceleration has become essential. Refer to (Silverman, 1986, 1994) for further information on elliptic curves.

Koblitz (1989) proposed that hyperelliptic curves of the arbitrary genus can replace elliptic curves in cryptosystems. However, this claim was falsified by Gaudry (2000), who showed that security is conversely proportional to the genus. Gaudry's work led the cryptographic studies to be focused on genus 1 and genus 2 cases specifically. Genus 2 hyperelliptic curves have been used in different areas. Lenstra Jr et al. (1993) incorporated genus 2 hyperelliptic curves for the factorization of B-smooth numbers, analogous to the Elliptic Curve Method (ECM) (Montgomery, 1987). Gaudry, Hess, and Smart (2002) pointed out that while hyperelliptic curves can help construct cryptosystems, they can also provide attacks on elliptic curve cryptosystems. The fastest results for pairing on hyperelliptic curves were provided by Barreto et al. (2007), defeating elliptic curves.

A genus 1 curve is an elliptic curve, and the points of 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, the Picard group of its divisors forms a group under the binary operation, namely divisor addition. A landmarking algorithm for divisor addition was introduced by Cantor (1987, §3, §4). The algorithm operates on Mumford's coordinates (Mumford, 1984), which is a polynomial representation of divisors and, consequently, makes heavy use of polynomial arithmetic. Therefore the algorithm is relatively inefficient for cryptographic applications. However, the algorithm is deterministic and works for arbitrary genus hyperelliptic curves. Explicit formulas are studied for hyperelliptic curves of genus 2 to dispose of the inefficient polynomial arithmetic. After much notable work on the topic, Gaudry (2007) proposed 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 (Bernstein, Chuengsatiansup, Lange, & Schwabe, 2014). A comprehensive explanation of the aspects of hyperelliptic curves can be found in the cornerstone book of Cassels and Flynn (1996). Refer to (Menezes, Zuccherato, & Wu, 1998) for a brief introduction.

Curve-based cryptosystems are actively being used in secure communication. It is of great importance to produce fast addition/scalar multiplication formulas for these cryptosystems. Once these fast formulas are implemented in a low-level programming language (i.e., assembly), cryptographic protocols can be constructed upon them to be used in secure communication channels. While ECC is widely being used in protocols, Hyperelliptic Curve Cryptosystems (HECC) is still in the development stage.

1.1. MOTIVATION

Hyperelliptic curves of genus 2 are promising for cryptographic applications. Although many sources make simple and understandable descriptions, a comprehensive interpretation of arithmetic on the curve is missing in the literature. The geometric interpretation is often neglected when describing rare cases. Cantor's algorithm provides a bird's-eye view of the work done in polynomial arithmetic. However, when the algorithm's operation is examined case-by-case, the picture in rare cases is revealed algebraically. Illustrating the geometric, polynomial arithmetic, and divisor arithmetic points of view, along with the connections in between, is one of the motivations of this work. This multidirectional interpretation seems necessary to support the advancements in this promising field.

Although not widely used yet, genus 2 hyperelliptic curves will soon find a place in cryptographic applications. One of the essential accelerators of this transition process is yielding fast group law formulas. In this sense, the most recent development is

Hisil and Costello's Jacobian coordinates. Unlike those previously offered, these coordinates have weights that naturally match the Jacobian, reducing the operation counts of divisor addition significantly. However, the adoption of Jacobian coordinates on such curves is restricted to common (Hisil & Costello, 2017) and a few degenerate cases (Hu, Lin, & Zhao, 2021). This leaves out the rare cases which do not contain a degenerate input/output. Therefore, previous studies cannot be considered conclusive, and a complete explicit algorithm for divisor addition on Jacobian coordinates seems necessary for a state-of-art software implementation.

To provide a guide and ready-to-use formulas for cryptographic applications are the motivations of this thesis.

1.2. AIMS AND OUTCOMES

The aims to achieve in this thesis are as follows:

- enhancing the group law on genus 2 hyperelliptic curves,
- associating different perspectives of the group law,
- reviewing how the formulas and algorithms for group law have evolved and accelerated in the literature over time,
- studying inversion-free formulas for divisor addition adopting different weights in projective coordinates,
- investigating different weighted coordinates that are missing in the literature.

The outcomes achieved with this thesis can be summarized as follows:

- a comprehensive interpretation of the complete group law on genus 2 hyperelliptic curves via
  - geometric construction,
  - divisor arithmetic,
  - polynomial arithmetic, and
  - explicit formulas,
- an inversion-free, complete, and explicit algorithm for divisor addition on genus 2 hyperelliptic curves on weighted Jacobian coordinates,
- scripts in Magma language that verify the complete group law algebraically,
- a script in Magma language that verifies the complete group law empirically,
- an article titled *"Complete Group Law for Genus 2 Jacobians on Jacobian*

Coordinates" has been submitted to the *Journal of Cryptographic Engineering* that passed the technical review.

1.3. LITERATURE REVIEW

Public-key cryptosystems can be constructed upon a finite abelian group if the DLP is hard for the group (Koblitz, 1989). ECC is based on the fact that the points of an elliptic curve form such an abelian group. As the Jacobian variety of a hyperelliptic curve also provides the necessary conditions, Koblitz (1989) indicated the potential of HECC. Hyperelliptic curves with genus greater than 1 provide the same level of security with a smaller base field in contrast to elliptic curves. This development gave rise to hopes that the key size could be reduced to much more efficient levels. However, a sub-exponential algorithm proposed by Adleman, DeMarrais, and Huang (1994) claimed that cryptosystems based on hyperelliptic curves of genus sufficiently high could be broken easily. The algorithm was based on heuristic assumptions, which were criticized by Flassenberg and Paulus (1999) for being unrealistic in where they proposed a sieving version of the algorithm. Müller, Stein, and Thiel (1999) offered a probabilistic algorithm that runs on subexponential time. The assumptions made by Adleman et al. (1994) were disposed of by Enge (2002). Gaudry (2000) introduced an algorithm similar to index-calculus and an experiment where a cryptosystem based on a genus 6 hyperelliptic curve is broken. He has shown that hyperelliptic curves of genus greater than 4 do not provide the desired security level and should be avoided. A faster algorithm for genus ≥ 3 is given by Thériault (2003), which is further advanced by Gaudry et al. (2007) that resulted in an algorithm asymptotically faster than Pollard's Rho method (Pollard, 1978).

Cantor (1987) introduced a deterministic algorithm for divisor addition on arbitrary genus hyperelliptic curves. Although the founding of Cantor's algorithm is a ground-breaking development, the heavy use of polynomial arithmetic makes it relatively inefficient for cryptographic applications. Adjustments to the algorithm are proclaimed in advance. Koblitz (1989) generalized the curve equation used in Cantor's algorithm to include characteristic 2. Smart (1999) improved the reduction step of the algorithm. Nagao (2000) proposed inversion-free division on polynomials and the computation of the extended greatest common divisor (GCD) with one inversion only.

The difficulty of polynomial arithmetic led to advances in explicit formulas for fixed genus hyperelliptic curves. Lange (2002b) reports in her thesis that Spallek (1994) proposed explicit formulas for the most common addition on genus 2 curves, later optimized by Gaudry and Harley (2000) for odd characteristic curves [1]. Harley's

[1] Last accessed: 14 February 2023, `http://cristal.inria.fr/~harley/hyper/adding`

approach is extended for even characteristics by Lange (2002b) in her thesis, where she also gave formulas for the rare cases. Matsuo, Chao, and Tsujii (2001) improved Harley's formulas, which were later sped up by Miyamoto (2002) and Takahashi (2002). Both Miyamoto and Takahashi exploited Montgomery's trick to cut the number of inversions to 1. Sugizaki, Matsuo, Chao, and Tsujii (2002) and Lange (2002a) advanced the work in (Miyamoto, 2002; Takahashi, 2002) for even characteristic independently. The explicit formulas for common addition and doubling in projective homogeneous coordinates are given by Lange (2002c). Later, Lange (2002d) produced the formulas for weighted coordinates, with 2 as the weight for the first two coordinates and 3 for the latter, and gave operation counts for different characteristics.

Hyperelliptic curves gained popularity with the advancements that indicate the analogous structure between ECC and HECC. Hess, Seroussi, and Smart (2001) presented the hyperelliptic versions of verifiably random curve construction and divisor compression. A better approach to divisor compression is proposed by Stahlke (2004). Avanzi (2004) showed that the performance of HECC is closer to ECC than it was thought to be. T. J. Wollinger (2004) achieved the best operation counts by applying various methods such as normalization, Karatsuba multiplication, and reduction and optimization of inversion-free formulas. In his thesis, T. J. Wollinger (2004, §6) also presents the software implementation of a hyperelliptic cryptosystem. While improving Cantor's and Harley's algorithms, T. Wollinger, Pelzl, and Paar (2005) stated that Cantor's algorithm could be more efficient than Harley's approach in some cases. A different approach is presented by Costello and Lauter (2012) to produce explicit formulas, where the new formulas for Cantor's composition step are derived by incorporating solving linear systems instead of polynomial arithmetic. Hisil and Costello (2017) extended the work (Costello & Lauter, 2012) by introducing Jacobian coordinates, reducing the operation counts significantly.

Degenerate divisors are proven to have secure usage in hyperelliptic cryptosystems (Katagi, Kitamura, Akishita, & Takagi, 2003). Although a chosen divisor is unlikely to be degenerate, in some cases, the divisor may be chosen degenerate on purpose for the sake of faster arithmetic. Duursma and Lee (2003) first mentioned using degenerate divisors for pairing computation on genus ≥ 2 hyperelliptic curves. The second argument of the pairing can be fixed (Costello & Stebila, 2010), which allows the use of a degenerate divisor in place of the second argument. Many studies exploited the use of degenerate divisors in pairing computation and achieved significant speed-up (Barreto et al., 2007; Katagi, Akishita, Kitamura, & Takagi, 2005; Katagi et al., 2003).

A recent work by Hu et al. (2021) solely analyzes the cases of addition where a degenerate divisor is present and gives the formulas in terms of the common sub-

`.text`

expressions proposed by Hisil and Costello (2017). The work of Hu et al. (2021) does not take care of the cases where neither the input divisors nor the output divisor is degenerate, but the common addition formulas do not apply. These cases occur because of joint supports of the input divisors and are all handled in the algorithm Lange (2002b) presents.

Apart from these developments, which are the main focus of this work, Gaudry (2007) 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 (Bernstein et al., 2014). However, some cryptographic constructions enforce the use of prime order genus 2 Jacobians, which do not allow a Kummer parameterization.

1.4. ORGANIZATION

The thesis is organized as follows. A background on genus 2 hyperelliptic curves and the group law will be given in Chapter 2 to form the necessary infrastructure for the upcoming chapters, including the properties of hyperelliptic curves, divisors, Mumford's representation, and Cantor's algorithm. Chapter 3 presents the 4-fold interpretation of divisor addition, along with the explicit formulas in affine coordinates for all possible cases. The inversion-free explicit formulas in Jacobian coordinates are going to be examined in Chapter 4. Finally, Chapter 6 will conclude the thesis with a summary of the work done. For the convenience of the reader, the computer-aided verification scripts are attached in the Appendix.

CHAPTER 2
PRELIMINARY

This chapter recalls basic definitions and facts about hyperelliptic curves. Throughout this section, 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

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

in $K[x, y]$ where $h(x), f(x) \in K[x]$, deg$(h) \leq g$, deg$(f) = 2g + 1$, and $f(x)$ is monic. The point set of H is denoted 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 a special point called the point at infinity. In the following sections, the notation $f'(x)$ and $f''(x)$ are used for the derivatives $\frac{df(x)}{dx}$ and $\frac{d^2f(x)}{dx^2}$, 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

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

The following sections will use the letter H to address such a curve.

2.2. DIVISORS

In order to present the group law on H, recalling basic definitions of the divisor theory is useful. 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 the zero divisor, denoted 0, with no nonzero n_P. The support of D is the set supp$(D) = \{P \in H(\bar{\mathbb{K}}) : n_P \neq 0\}$, and the degree of D is given by deg$(D) = \sum_{P \in H(\bar{K})} n_P$. A divisor of a function f is given by $(f) = \sum_{P \in H(\bar{K})}$ ord$_f(P)$ where 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 D with $\deg(D) \leq g$.

Throughout this thesis, the zero, degree 1, and degree 2 divisors will be referred to as trivial, degenerate, and non-degenerate divisors, respectively.

The set of all divisors are denoted as $\text{Div}_{\bar{K}}(H)$; the degree zero divisors as $\text{Div}^0_{\bar{K}}(H)$; and the principal divisors (divisors of a function) as $\text{Prin}_{\bar{K}}(H)$. These sets all form groups that satisfy the relation $\text{Prin}_{\bar{K}}(H) \subseteq \text{Div}^0_{\bar{K}}(H) \subseteq \text{Div}_{\bar{K}}(H)$. The quotient group $\text{Div}^0_{\bar{K}}(H)/\text{Prin}_{\bar{K}}(H)$ is called the Picard group and is denoted by $\text{Pic}_{\bar{K}}(H)$. Picard group is a divisor class group and isomorphic to the Jacobian variety of H, denoted by $J(H)$. Two divisors $D_1, D_2 \in J(H)$ are equivalent, denoted $D_1 \sim D_2$, if $D_1 - D_2 \in \text{Prin}_{\bar{K}}(H)$. The critical observation about the Jacobian is that each of its cosets 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 the Jacobian of a genus 2 hyperelliptic can be represented using the Mumford representation (Mumford, 1984) 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's 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 in $H(K)$ but not $H(\bar{K})$. 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 $P_2 = (x_2, y_2) \in H(\bar{K})$, the notation $D = [x^2 + qx + r, sx + t]$ is used. Here, assuming $x_1 \neq x_2$,

$$(q, r, s, t) = \left(-x_1 - x_2,\ x_1 x_2,\ \frac{y_1 - y_2}{x_1 - x_2},\ \frac{x_1 y_2 - x_2 y_1}{x_1 - x_2} \right), \tag{2.2}$$

otherwise

$$(q, r, s, t) = \left(-2x_1,\ x_1^2,\ \frac{5x_1^4 + 3a_3 x_1^2 + 2a_2 x_1 + a_1}{2y_1},\ \frac{-3x_1^5 - a_3 x_1^3 + a_1 x_1 + 2a_0}{2y_1} \right). \tag{2.3}$$

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. GROUP LAW

The point set of elliptic curves, along with the point at infinity, forms a group; however, this does not hold for hyperelliptic curves of genus greater than 1. Consequently, the group law is defined on the divisor class group $\mathrm{Pic}_{\bar{K}}(H)$ for genus $g \geq 2$ curves.

A function f that intersects the curve at the points in $\mathrm{supp}(D_1)$ and $\mathrm{supp}(D_2)$ also intersects the curve at the negatives of the points in $\mathrm{supp}(D_1 + D_2)$. Negating the found intersection points completes the addition. The operation is geometrically represented in Figure 2.1. Fortunately, the entire formula can be written in terms of Mumford's

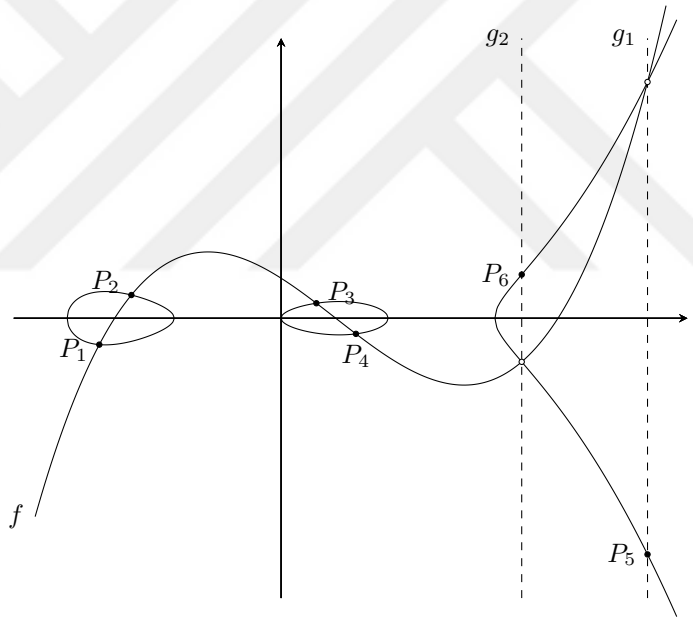

Figure 2.1. Divisor Addition

representation, which means the coordinates of the points in the supports need not be known explicitly.

The divisor arithmetic behind the addition is analogous to the geometric interpretation. Two divisors, D_1 and D_2, in $\mathrm{Pic}_{\bar{K}}(H)$ are equal if their sum/difference can be written as a principal divisor. Thus, the function f and the vertical lines g_1 and g_2 form the

principal divisors that can be freely added to the system of equations.

$$
\begin{aligned}
D_1 &= (P_1) + (P_2) - 2(\mathcal{O}) \\
D_2 &= (P_3) + (P_4) - 2(\mathcal{O}) \\
(f) &= (P_1) + (P_2) + (P_3) + (P_4) + (-P_5) + (-P_6) - 6(\mathcal{O})
\end{aligned}
$$

$$
\begin{aligned}
D_1 + D_2 &\sim D_1 + D_2 - (f) \\
&\sim -(-P_5) - (-P_6) + 2(\mathcal{O})
\end{aligned}
$$

$$
\begin{aligned}
(g_1) &= (P_5) + (-P_5) - 2(\mathcal{O}) \\
(g_1) &= (P_6) + (-P_6) - 2(\mathcal{O})
\end{aligned}
$$

$$
\begin{aligned}
D_1 + D_2 &\sim D_1 + D_2 - (f) + (g_1) + (g_2) \\
&\sim (P_5) + (P_6) - 2(\mathcal{O})
\end{aligned}
$$

Another way to interpret divisor addition is through using polynomial arithmetic in Cantor's algorithm. The algorithm operates on the polynomials in Mumford's representation that define the divisors and the polynomials that define the equation of the curve. Cantor's algorithm is going to be detailed in the next section.

2.4. CANTOR'S ALGORITHM

Cantor (1987, §3, §4) introduced an algorithm to compute divisor addition on hyperelliptic curves, analogous to class group arithmetic, which is generalized and presented later by Koblitz (1989); Menezes et al. (1998). Cantor's algorithm mostly depends on the theory of class groups of quadratic fields and the ideal composition methods proposed by Daniel (1971), see (Lang, 1959). The algorithm operates polynomial arithmetic on quadratic forms and works for hyperelliptic curves of arbitrary genus. There are two stages of the algorithm; the composition stage (Figure 2.2 lines 1-4) computes the semi-reduced divisor that is equal to the addition of the inputs; the reduction stage (Figure 2.2 lines 5-9) computes the related unique reduced divisor. For further information, refer the reader to the appendix of (Koblitz, Menezes, Wu, & Zuccherato, 1998).

At the composition stage, all the roots are brought together, and then, if there are common roots, the necessary elimination is done to prevent duplicates. The composition phase exploits the Chinese Remainder Theorem (CRT) to construct of the semi-reduced divisor. The reduction phase is based on Gauss's quadratic form reduction algorithm

Input: Reduced divisors $D_1 = [u_1, v_1], D_2 = [u_2, v_2] \in Jac(H)$ and hyperelliptic curve $H(K) : y^2 + h(x)y = f(x)$ of genus g
Output: Reduced divisor $D_3 = [u, v]$ where $D_3 \sim D_1 + D_2$

1 Compute d_1, e_1, e_2 where $d_1 = \texttt{gcd}(u_1, u_2) = e_1u_1 + e_2u_2$
2 Compute d, c_1, c_2 where $d = \texttt{gcd}(d_1, v_1 + v_2 + h) = c_1d_1 + c_2(v_1 + v_2 + h)$
3 Let $s_1 = c_1e_1$, $s_2 = c_1e_2$ and $s_3 = c_2$ such that
  $d = s_1u_1 + s_2u_2 + s_3(v_1 + v_2 + h)$
4 $u = \dfrac{u_1u_2}{d^2},\ v = \dfrac{s_1u_1v_2 + s_2u_2v_1 + s_3(v_1v_2 + f)}{d} \mod u$
5 $u' = \dfrac{f - vh - v^2}{u},\ v' = (-h - v) \mod u'$
6 **if** $\texttt{deg}(u') > g$ **then**
7 $u \leftarrow u', v \leftarrow v'$
8 Go to step 5
9 **end if**
10 $u' \leftarrow \texttt{monic}(u')$
11 **return** u', v'

Figure 2.2. Cantor's Algorithm

and is applied to the semi-reduced divisor until the degree falls below the genus and the unique reduced representative is found.

This algorithm is used to verify the correctness of the explicit formulas presented in Section 3.1. Since the computations involving these verifications are tedious, carrying them out succinctly with the help of computer algebra is preferred rather than a pages-long algebraic investigation. Now the complete and explicit group law is presented in Chapter 3.

CHAPTER 3
DIVISOR ADDITION

Beforehand, the basic facts referenced in the algorithm's body are given.

Lemma 3.1. *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. Simply Mumford's composition. □

Lemma 3.2. *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 3.3. *No degenerate divisor $D \in J(H)$ exists 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, $2D = (-P) - (\mathcal{O})$. However, by Lemma 3.1, $2D$ must be non-degenerate while $(-P) - (\mathcal{O})$ is clearly degenerate. This is a contradiction by Lemma 3.2. Therefore, D must be non-degenerate. □

In the remainder of this section, two input divisors, D_1, D_2, and an output divisor $D_3 = D_1 + D_2$ are defined, and the complete formulas for D_3 are examined. The output divisor is set as $D_3 = [x - x_5, y_5]$ when degenerate and $D_3 = [x^2 + q_3x + r_3, s_3x + t_3]$ when non-degenerate. The cases are interpreted from 4 perspectives: geometric construction, divisor arithmetic, polynomial arithmetic of Cantor's algorithm, and explicit formulas. Often the polynomial arithmetic and geometric interpretation are omitted when it is trivial or complicated to be presented here.

3.1. INTERPRETATION AND EXPLICIT FORMULAS

3.1.1. TRIVIAL INPUTS

Here, the output is investigated when at least one of the input divisors is the identity, namely the degree 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).

3.1.2. 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, it is investigated 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 $y_1 = -y_3$: Then, $P_1 = -P_3$. The direct addition of the divisors gives the degree zero divisor $(P_1) + (-P_1) - 2(\mathcal{O})$, which is neither reduced nor semi-reduced. However, since the divisors are on the Jacobian, any principal divisor can be added to the sum to reach the unique reduced representative. The divisor arithmetic is shown below.

$$\begin{array}{lcl} D_1 + D_2 & = & (P_1) + (-P_1) - 2(\mathcal{O}) \\ (g) & = & (P_1) + (-P_1) - 2(\mathcal{O}) \\ - & & \\ \hline D_1 + D_2 & \sim & 0 \text{ (The Zero Divisor)} \end{array}$$

Such principal divisor (g) exists, and function g corresponds to a vertical line, as seen in Figure 3.1.

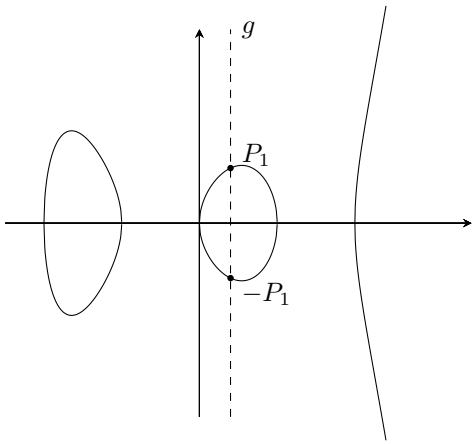

Figure 3.1. $D_1 \sim (P_1) - (\mathcal{O}), \quad D_2 \sim (-P_1) - (\mathcal{O})$

Cantor's algorithm gives the same result via polynomial arithmetic, summarized below.

$$d_1 = \gcd(u_1, u_2) = \gcd(x - x_1, x - x_1) = x - x_1 \qquad \text{(Cantor step 1)}$$
$$d = \gcd(d_1, v_1 + v_2) = \gcd(x - x_1, 0) = x - x_1 \qquad \text{(Cantor step 2)}$$
$$u = \frac{u_1 u_2}{d^2} = \frac{(x - x_1)(x - x_1)}{(x - x_1)^2} = 1 \qquad \text{(Cantor step 4)}$$
$$v = 0 \text{ (After some more complicated calculations)}$$

Thus, $D_3 = 0$.

- Case $y_1 = y_3$: So, $P_1 = P_3$. Therefore, $D_1 = D_2$ and $D_3 = 2D_1 = 2(P_1) - 2(\mathcal{O}) = [x^2 + q_3 x + r_3,\ s_3 x + t_3]$ is non-degenerate and can be computed with Mumford's composition Equation (2.3). [Script 12] in the Appendix verifies the correctness of this formula.

- Case $x_1 \neq x_3$: The input divisors are disjoint and $D_3 = [x^2 + q_3 x + r_3,\ s_3 x + t_3]$ can be computed with Mumford's composition Equation (2.2). [Script 13] 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 3.1. Note that the case where $x_1 = x_3$ and $y_1 = y_3 = 0$ (special point doubling) is handled by "Case $y_1 = -y_3$".

3.1.3. DEGENERATE/NON-DEGENERATE INPUTS

The addition of a non-degenerate and a degenerate divisor is the next step. Assume that the inputs are swapped, if necessary, in order to have D_1 and D_2 degenerate and non-degenerate, respectively. Let $D_1 = [x - x_1, y_1]$ and $D_2 = [u_2(x), v_2(x)] = [x^2 + q_2 x + r_2,\ s_2 x + t_2]$ be divisors in $J(H)$. Set $P_1 = (x_1, y_1) \in \mathbb{H}(K) - \{\mathcal{O}\}$ such that $D_1 = (P_1) - (\mathcal{O})$.

- Case $u_2(x_1) = 0$: Since $x_1 \in K$ is a root of the polynomial $u_2(x)$, $u_2(x)$ must have the roots $x_3, x_4 \in K$ satisfying $(x_3 - x_1)(x_4 - x_1) = 0$, though x_3 and x_4 are not known explicitly. Without loss of generality, we assume $x_3 = x_1$. Now, y_3, x_4, and y_4 can be computed as $y_3 = v_2(x_1)$, $x_4 = -q_2 - x_3$, and $y_4 = v_2(x_4)$. Clearly, $x_1, y_1, x_3, y_3, x_4, y_4 \in K$. Before moving forward to investigate the following subcases, it is important to note that $P_3 = (x_3, y_3)$ and $P_4 = (x_4, y_4)$ are in $H(K) - \{\mathcal{O}\}$ and $D_2 = (P_3) + (P_4) - 2(\mathcal{O})$.

  - Case $y_1 = -y_3$: Then, $P_1 = -P_3$. The opposite points cancel each other out when the vertical line that passes through them is added to the sum as a principal divisor. The arithmetic is shown below.

$$
\begin{array}{lcl}
D_1 + D_2 & = & (P_1) + (-P_1) + (P_4) - 3(\mathcal{O}) \\
(g) & = & (P_1) + (-P_1) - 2(\mathcal{O}) \\
- & & \\
\hline
D_1 + D_2 & \sim & (P_4) - (\mathcal{O})
\end{array}
$$

The composition stage of Cantor's algorithm, likewise, cancels out points with their negatives. For this case, it is known that $x_1 = x_3$, $y_1 = -y_3$ and $v_1 = y_1$, then $(x - x_1)$ in u vanishes. This is shown as follows:

$$
\begin{aligned}
d_1 &= \text{gcd}(u_1, u_2) = \text{gcd}((x - x_1), (x - x_1)(x - x_4)) = (x - x_1) && \text{(Cantor step 1)} \\
d &= \text{gcd}(d_1, v_1 + v_2) = \text{gcd}(x - x_1, y_1 + s_2 x + t_2) && \text{(Cantor step 2)} \\
&= \text{gcd}\left(x - x_1, \frac{-y_1 - y_4}{x_1 - x_4}(x - x_1)\right) \\
&= \text{gcd}(x - x_1, s_2(x - x_1)) = x - x_1 \\
u &= \frac{u_1 u_2}{d^2} = \frac{(x - x_1)^2 (x - x_4)}{(x - x_1)^2} = (x - x_4). && \text{(Cantor step 4)}
\end{aligned}
$$

Similarly, v reduces to y_4 and the degenerate output is $D_3 = (P_4) - (\mathcal{O}) = [x - x_4, y_4]$.

– Case $y_1 = y_3$: Here $P_1 = P_3$ and moving forward with investigating the rare case $P_1 = P_3 = P_4$.

  * Case $x_1 = x_4$: Here, $P_1 = \pm P_4$. Now, $P_1 = -P_4$ is not possible, because otherwise $D_2 = (P_1) + (-P_1) - 2(\mathcal{O})$ would be non-reduced. Therefore, $P_1 = P_3 = P_4$, and the divisor arithmetic behind the calculation of D_3 consists of forming the principal divisor with function f that passes through P_1 with order 3. The function f intersects the curve at 2 other points, $-P_5$ and $-P_6$, which are the negatives of the points in the resulting divisor's support. Then, what remains is only to negate $-P_5$ and $-P_6$ with the vertical lines g_1 and g_2.

$$
\begin{array}{lcl}
D_1 + D_2 & = & 3(P_1) - 3(\mathcal{O}) \\
(f) & = & 3(P_1) + (-P_5) + (-P_6) - 5(\mathcal{O}) \\
- & & \\
\hline
 & = & -(-P_5) - (-P_6) + 2(\mathcal{O}) \\
(g_1) & = & (P_5) + (-P_5) - 2(\mathcal{O}) \\
(g_2) & = & (P_6) + (-P_6) - 2(\mathcal{O}) \\
+ & & \\
\hline
D_1 + D_2 & \sim & (P_5) + (P_6) - 2(\mathcal{O})
\end{array}
$$

The explicit formula for the computation of $D_3 = 3(P_1) - 3(\mathcal{O}) = 3D_1$ is given

in (Hu et al., 2021, §3.2) as

$$\begin{aligned} q_3 &= 3x_1 - A^2, \\ r_3 &= a_3 - 2A \cdot B + 3x_1(q_3 - x_1), \\ s_3 &= A \cdot q_3 - B, \\ t_3 &= A \cdot r_3 - C, \end{aligned} \tag{3.1}$$

where the common subexpressions A, B, and C are given as $(2y_1^2 f''(x_1) - f'(x_1)^2)/(8y_1^3)$, $f'(x_1)/(2y_1) - 2A \cdot x_1$, and $y_1 - A \cdot x_1^2 - B \cdot x_1$, respectively. Note that $D_1 = (P_1) - (\mathcal{O})$ cannot have order 3[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 18] 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. The addition of the divisors gives the semi-reduced divisor $D_3 = 2(P_1) + (P_4) - 3(\mathcal{O})$ which can be reduced to its final form with the help of a function f. The function f intersects the curve at P_1 twice and P_4 once, and it forms a principal divisor. It intersects the curve in two other points $-P_5$ and $-P_6$, which can be negated with the vertical lines g_1 and g_2.

$$\begin{array}{lcl} D_1 + D_2 & = & 2(P_1) + (P_4) - 3(\mathcal{O}) \\ (f) & = & 2(P_1) + (P_4) + (-P_5) + (-P_6) - 5(\mathcal{O}) \\ - & & \\ \hline & = & -(-P_5) - (-P_6) + 2(\mathcal{O}) \\ (g_1) & = & (P_5) + (-P_5) - 2(\mathcal{O}) \\ (g_2) & = & (P_6) + (-P_6) - 2(\mathcal{O}) \\ + & & \\ \hline D_1 + D_2 & \sim & (P_5) + (P_6) - 2(\mathcal{O}) \end{array}$$

The functions are represented geometrically in Figure 3.2.

For the explicit formula, the addition can be split into two parts: the doubling of $D_1 = (P_1) - (\mathcal{O})$ and the accumulation of $(P_4) - (\mathcal{O})$ on $2D_1$. First, the divisor $2(P_1) - 2(\mathcal{O}) = [x^2 + qx + r, sx + t]$ is computed as in (2.3). Note that the denominator y_1 in (2.3) does not vanish since the case $y_1 = 0$ is again handled in "Case $y_1 = -y_3$". Then the resulting non-degenerate divisor is calculated with

[1]This is assumed in (Hu et al., 2021, §3.2). The proof can be found in Lemma 3.3.

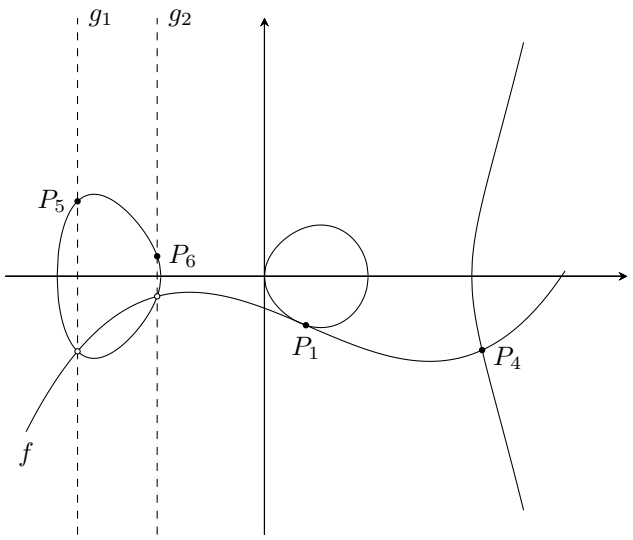

Figure 3.2. $D_1 \sim (P_1) - (\mathcal{O}), \quad D_2 \sim (P_1) + (P_4) - 2(\mathcal{O})$

the following formula,

$$\begin{aligned} q_3 &= x_4 - q - A^2, \\ r_3 &= a_3 + q^2 - r - A \cdot (B + s) + x_4 q_3, \\ s_3 &= A \cdot q_3 - B, \\ t_3 &= A \cdot r_3 - C, \end{aligned} \tag{3.2}$$

where the common subexpressions A, B, and C are given as $(y_4 - sx_4 - t)/(x_4^2 + qx_4 + r)$, $s + q \cdot A$, and $t + r \cdot A$, respectively. Notice that the denominator of A vanishes only when P_4 is in the support of divisor $2(P_1) - 2(\mathcal{O})$, which contradicts the assumption $x_1 \neq x_4$. [Script 22] in the Appendix verifies the correctness of this formula.

- Case $u_2(x_1) \neq 0$: In this case, the divisors are disjoint, and a function f passes through P_1, P_3, and P_4 once.

$$\begin{array}{lcl} D_1 + D_2 & = & (P_1) + (P_3) + (P_4) - 3(\mathcal{O}) \\ (f) & = & (P_1) + (P_3) + (P_4) + (-P_5) + (-P_6) - 5(\mathcal{O}) \\ - & & \\ \hline & = & -(-P_5) - (-P_6) + 2(\mathcal{O}) \\ (g_1) & = & (P_5) + (-P_5) - 2(\mathcal{O}) \\ (g_2) & = & (P_6) + (-P_6) - 2(\mathcal{O}) \\ + & & \\ \hline D_1 + D_2 & \sim & (P_5) + (P_6) - 2(\mathcal{O}) \end{array}$$

The function f and the vertical lines g_1 and g_2 are represented in Figure 3.3.

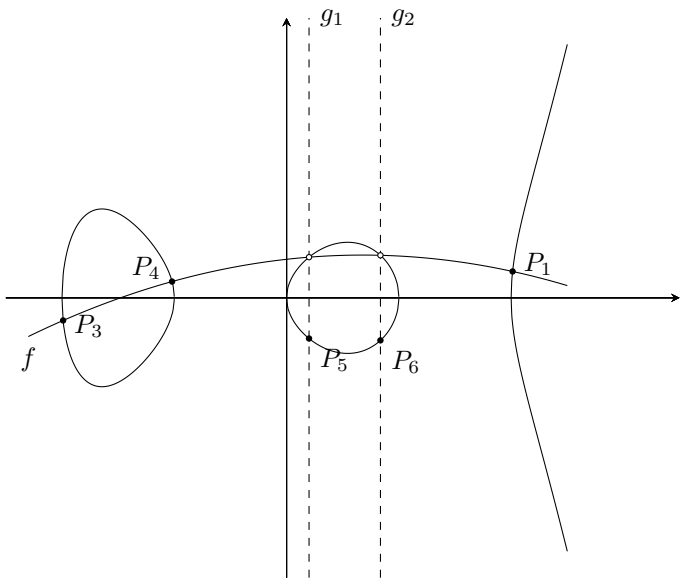

Figure 3.3. $D_1 \sim (P_1) - (\mathcal{O}), \quad D_2 \sim (P_3) + (P_4) - 2(\mathcal{O})$

Here, the formula in (Hu et al., 2021, §3.1) applies, which is a general form of (3.2). Then,

$$\begin{aligned} q_3 &= x_1 - q_2 - A^2, \\ r_3 &= a_3 + q_2^2 - r_2 - A \cdot (B + s_2) + x_1 q_3, \\ s_3 &= A \cdot q_3 - B, \\ t_3 &= A \cdot r_3 - C, \end{aligned} \tag{3.3}$$

where the common subexpressions A, B, and C are given as $(y_1 - s_2 x_1 - t_2)/(x_1^2 + q_2 x_1 + r_2)$, $s_2 + q_2 \cdot A$, and $t_2 + r_2 \cdot A$, respectively. Note that the denominator $x_1^2 + q_2 x_1 + r_2 = u_2(x_1) \neq 0$. [Script 15] in the Appendix verifies the correctness of this formula.

Finally, recall that the addition of a non-degenerate divisor with a degenerate divisor cannot give the zero divisor by Lemma 3.2.

3.1.4. NON-DEGENERATE/NON-DEGENERATE INPUTS

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

- Case $u_1 = u_2$: This implies that $\{x_1, x_2\} = \{x_3, x_4\}$, though x_i and y_i are not known explicitly. Moving forward by investigating v_1 and v_2.
  - Case $v_1 = -v_2$: Now, 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 $\{y_1, y_2\} = \{-y_3, -y_4\}$. Without loss of generality we may assume $P_1 = -P_3$ and $P_2 = -P_4$. With the help of vertical lines g_1 and g_2,

$$\begin{array}{lcl} D_1 + D_2 & = & (P_1) + (-P_1) + (P_2) + (-P_2) - 4(\mathcal{O}) \\ (g_1) & = & (P_1) + (-P_1) - 2(\mathcal{O}) \\ (g_2) & = & (P_2) + (-P_2) - 2(\mathcal{O}) \\ - & & \\ \hline D_1 + D_2 & \sim & 0 \text{ (The Zero Divisor)} \end{array}$$

it results in $D_3 \sim 0$.

– Case $v_1 = v_2$: Similarly, 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. The common subexpressions A, B, and C are defined for doubling as in (Hisil & Costello, 2017, §5).

$$\begin{aligned} A &= ((q_1^2 - 4r_1 + a_3)q_1 - a_2 + s_1^2)(q_1 s_1 - t_1) + (3q_1^2 - 2r_1 + a_3)r_1 s_1, \\ B &= 2(q_1 s_1 - t_1)t_1 - 2r_1 s_1^2, \\ C &= ((q_1^2 - 4r_1 + a_3)q_1 - a_2 + s_1^2)s_1 + (3q_1^2 - 2r_1 + a_3)t_1. \end{aligned} \tag{3.4}$$

The doubling formula in (Hisil & Costello, 2017, §5) is not defined for two cases; when the output is degenerate and when the denominator B vanishes. The approach in (Lange, 2002b, §3.1,§3.4) properly handles each case, but it requires the use of polynomial arithmetic. However, to eliminate polynomial arithmetic and carry out the computation solely with field arithmetic is aimed here. Therefore, it is required to algebraically pinpoint the cases where the output is degenerate or non-degenerate. These cases are given as follows.

∗ Case $B = 0$: B is equal to $-2y_1y_2$ which corresponds to the resultant$(h + 2v_1, u_1)$ calculated in (Lange, 2002b, §3.1). Now, $B = 0$ induces at least one of the points in the support is a special point that gives the identity when doubled. Without loss of generality, we may assume P_2 is special. This indicates that there exists a vertical line g that intersects P_2 twice.

$$\begin{array}{lcl} D_1 + D_2 & = & 2(P_1) + 2(P_2) - 4(\mathcal{O}) \\ (g) & = & 2(P_2) - 2(\mathcal{O}) \\ - & & \\ \hline D_1 + D_2 & \sim & 2(P_1) - 2(\mathcal{O}) \end{array}$$

Since P_2 is special, so that $y_2 = 0$, x_2 can be computed by explicitly calculating $\gcd(u_1, v_1) = x - x_1$ as in (Lange, 2002b, §3.1). Then, $x_2 = -t_1/s_1$, $x_1 = -q_1 - x_2$ and $y_1 = v_1(x_1)$. Finally, $D_3 = 2(P_1) - 2(\mathcal{O})$ can be computed with the composition formula (2.3).

* 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 (Lange, 2002b) 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, the common subexpression C is investigated next.

  · Case $C = 0$: As stated in (Hu et al., 2021, §3.3), $C = 0$ implies a degenerate output since the coefficient of the degree 3 term in the function $f : y = dx^3 + ax^2 + bx + c$ is equal to $-C/B$. C being zero annihilates the degree 3 term, and f intersects the curve in exactly one more point instead of two.

$$
\begin{array}{lcl}
D_1 + D_2 & = & 2(P_1) + 2(P_2) - 4(\mathcal{O}) \\
(f) & = & 2(P_1) + 2(P_2) + (-P_5) - 5(\mathcal{O}) \\
- & & \\
\hline
 & = & -(-P_5) + (\mathcal{O}) \\
(g) & = & (P_5) + (-P_5) - 2(\mathcal{O}) \\
+ & & \\
\hline
D_1 + D_2 & \sim & (P_5) - (\mathcal{O})
\end{array}
$$

The functions f and g are represented geometrically in Figure 3.4.

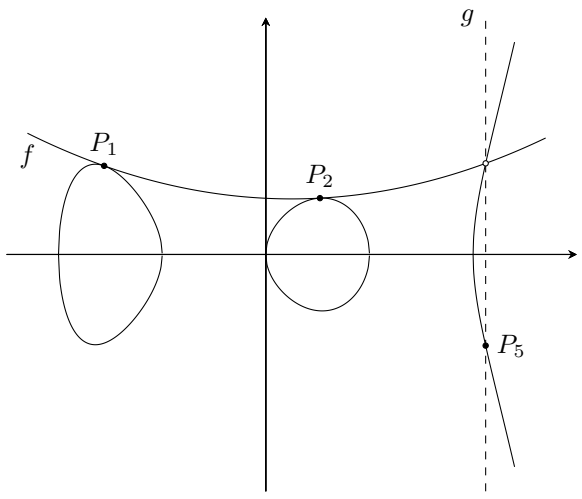

Figure 3.4. $D_1 \sim (P_1) + (P_2) - 2(\mathcal{O}), \quad D_2 \sim (P_1) + (P_2) - 2(\mathcal{O}), \quad C = 0$

The corrected version of the formula in (Hu et al., 2021, §3.3) is as follows [2]

$$\begin{aligned} x_5 &= 2q_1 + \frac{A^2}{B^2}, \\ y_5 &= \left(\frac{A}{B}(q_1 + x_5) - s_1\right) x_5 + \left(\frac{A}{B} r_1 - t_1\right). \end{aligned} \tag{3.5}$$

[Script 14] in the Appendix verifies the correctness of this formula.

· Case $C \neq 0$: This is the common case where the function intersects the curve at two more points.

$$\begin{array}{lcl} D_1 + D_2 & = & 2(P_1) + 2(P_2) - 4(\mathcal{O}) \\ (f) & = & 2(P_1) + 2(P_2) + (-P_5) + (-P_6) - 6(\mathcal{O}) \\ - & & \\ \hline & = & -(-P_5) - (-P_6) + 2(\mathcal{O}) \\ (g_1) & = & (P_5) + (-P_5) - 2(\mathcal{O}) \\ (g_2) & = & (P_6) + (-P_6) - 2(\mathcal{O}) \\ + & & \\ \hline D_1 + D_2 & \sim & (P_5) + (P_6) - 2(\mathcal{O}) \end{array}$$

The operation can be seen geometrically in Figure 3.5.

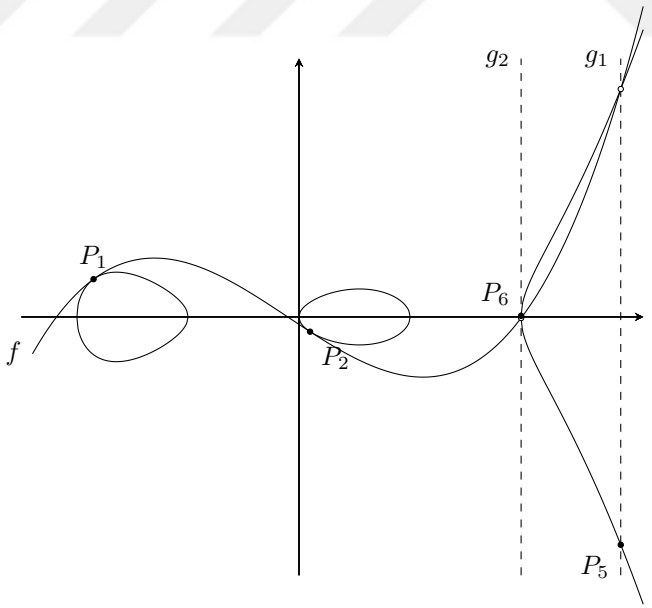

Figure 3.5. $D_1 \sim (P_1) + (P_2) - 2(\mathcal{O}), \quad D_2 \sim (P_1) + (P_2) - 2(\mathcal{O})$

Here, the doubling formula in (Hisil & Costello, 2017, §5) applies for the non-degenerate output D_3. The formulas are reproduced here to keep the text self-contained; see (Hisil & Costello, 2017, §5, §6) for operation counts and

[2] Incorrect x_3 is replaced with x_5.

further information.

$$
\begin{aligned}
q_3 &= 2\frac{A}{C} - \frac{B^2}{C^2}, \\
r_3 &= \frac{A^2}{C^2} + 2q_1\frac{B^2}{C^2} - 2s_1\frac{B}{C}, \\
s_3 &= (r_1 - r_3)\frac{C}{B} - q_3(q_1 - q_3)\frac{C}{B} + (q_1 - q_3)\frac{A}{B} - s_1, \\
t_3 &= (r_1 - r_3)\frac{A}{B} - r_3(q_1 - q_3)\frac{C}{B} - t_1.
\end{aligned}
\tag{3.6}
$$

[Script 16] 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, the equality $\{y_1, y_2\} = \{\pm y_3, \pm y_4\}$ holds. This results in 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 $P_1 = P_3$ and $P_2 = -P_4$. Then, there exists a vertical line g_1 that allows us to cancel P_2 with $-P_2$, and a function f that doubles P_1.

$$
\begin{array}{lcl}
D_1 + D_2 & = & (P_1) + (P_2) + (P_1) + (-P_2) - 4(\mathcal{O}) \\
(g_1) & = & (P_2) + (-P_2) - 2(\mathcal{O}) \\
- & & \\
\hline
 & = & 2(P_1) - 2(\mathcal{O}) \\
(f) & = & 2(P_1) + (-P_5) + (-P_6) - 4(\mathcal{O}) \\
- & & \\
\hline
 & = & -(-P_5) - (-P_6) + 2(\mathcal{O}) \\
(g_2) & = & (P_5) + (-P_5) - 2(\mathcal{O}) \\
(g_3) & = & (P_6) + (-P_6) - 2(\mathcal{O}) \\
+ & & \\
\hline
D_1 + D_2 & \sim & (P_5) + (P_6) - 2(\mathcal{O})
\end{array}
$$

Fortunately, x_1, y_1 can be computed with $x_1 = (t_1 - t_2)/(s_2 - s_1)$ and $y_1 = v_1(x_1)$ explicitly as in (Lange, 2002b, §3.1). The doubling of P_1 can be done with the formula (2.3).

• Case $u_1 \neq u_2$: Here, the supports of D_1 and D_2 are not identical. Thus, next are the addition formulas. Let $D_1 = [x^2 + q_1x + r_1, s_1x + t_1] = (P_1) + (P_2) - 2(\mathcal{O})$ and $D_2 = [x^2 + q_2x + r_2, s_2x + t_2] = (P_3) + (P_4) - 2(\mathcal{O})$. The common subexpressions

A, B, and C are defined as in (Hisil & Costello, 2017, §5),

$$\begin{aligned} A &= (t_1 - t_2)(q_2(q_1 - q_2) - (r_1 - r_2)) - r_2(q_1 - q_2)(s_1 - s_2), \\ B &= (r_1 - r_2)(q_2(q_1 - q_2) - (r_1 - r_2)) - r_2(q_1 - q_2)^2, \\ C &= (q_1 - q_2)(t_1 - t_2) - (r_1 - r_2)(s_1 - s_2). \end{aligned} \tag{3.7}$$

As before, the cases where the addition formula in (Hisil & Costello, 2017, §5) is not defined need to be detected first. The common subexpressions B and C are investigated for that purpose. The reason for concentrating on B is illuminated when B is written 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 (Lange, 2002b, §3.1) and becomes zero when $\{P_1, P_2\} \cap \{\pm P_3, \pm P_4\} \neq \emptyset$, which means that the supports of the input divisors are joint. When $B = 0$, the common addition formula collapses since B is in the denominator. However, the intersection of the supports gives a clue about the explicit formulas for the coordinates of the points P_1, P_2, P_3, P_4.

– 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 (Lange, 2002b, §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.

$$\begin{array}{lcl} D_1 + D_2 & = & (P_1) + (P_2) + (-P_1) + (P_4) - 4(\mathcal{O}) \\ (g_1) & = & (P_1) + (-P_1) - 2(\mathcal{O}) \\ - & & \\ \hline & = & (P_2) + (P_4) - 2(\mathcal{O}) \\ (f) & = & (P_2) + (P_4) + (-P_5) + (-P_6) - 4(\mathcal{O}) \\ - & & \\ \hline & = & -(-P_5) - (-P_6) + 2(\mathcal{O}) \\ (g_2) & = & (P_5) + (-P_5) - 2(\mathcal{O}) \\ (g_3) & = & (P_6) + (-P_6) - 2(\mathcal{O}) \\ + & & \\ \hline D_1 + D_2 & \sim & (P_5) + (P_6) - 2(\mathcal{O}) \end{array}$$

After eliminating P_1 with $-P_1$ (with g_1), what remains is the composition of P_2 and P_4 (with f), which can be calculated with the formula in (2.2).

∗ Case $y_1 = y_3$: Here $P_1 = P_3$ and the addition consist of the doubling of P_1 and

the addition of P_2 and P_4.

$$\begin{array}{lcl}
D_1 + D_2 & = & (P_1) + (P_2) + (P_1) + (P_4) - 4(\mathcal{O}) \\
(f) & = & 2(P_1) + (P_2) + (P_4) + (-P_5) + (-P_6) - 6(\mathcal{O}) \\
- & & \\
\hline
 & = & -(-P_5) - (-P_6) + 2(\mathcal{O}) \\
(g_1) & = & (P_5) + (-P_5) - 2(\mathcal{O}) \\
(g_2) & = & (P_6) + (-P_6) - 2(\mathcal{O}) \\
+ & & \\
\hline
D_1 + D_2 & \sim & (P_5) + (P_6) - 2(\mathcal{O})
\end{array}$$

Explicitly, $P_1 = (x_1, y_1)$ can be doubled with the formula in (2.3), and $P_2 = (x_2, y_2)$ and $P_4 = (x_4, y_4)$ can be added one by one using the procedure in Section 3.1.3.

– Case $B \neq 0$: Here, the supports of the divisors are disjoint. Next, the case of degenerate output is investigated. An accidental cancellation occurs in the addition formulas of (Lange, 2002b, §3.2) when the output is degenerate, just as in doubling. In (Hu et al., 2021, §3.3), it is stated that $C = 0$ implies a degenerate output since the coefficient $d = -C/B = 0$ in the function $f : y = dx^3 + ax^2 + bx + c$ annihilates the degree 3 term.

  ∗ Case $C = 0$: The function f intersects the curve at one more point instead of two.

$$\begin{array}{lcl}
D_1 + D_2 & = & (P_1) + (P_2) + (P_3) + (P_4) - 4(\mathcal{O}) \\
(f) & = & (P_1) + (P_2) + (P_3) + (P_4) + (-P_5) - 5(\mathcal{O}) \\
- & & \\
\hline
 & = & -(-P_5) + (\mathcal{O}) \\
(g) & = & (P_5) + (-P_5) - 2(\mathcal{O}) \\
+ & & \\
\hline
D_1 + D_2 & \sim & (P_5) - (\mathcal{O})
\end{array}$$

See Figure 3.6 for a geometric interpretation of the case.

The corrected version of the formula in (Hu et al., 2021, §3.3) is as follows, [3]

$$\begin{aligned}
x_5 &= (q_1 + q_2) + \frac{A^2}{B^2}, \\
y_5 &= x_5(q_1 + x_5)\frac{A}{B} - x_5 s_1 + r_1 \frac{A}{B} - t_1.
\end{aligned} \tag{3.8}$$

[Script 12] in the Appendix verifies the correctness of this formula.

[3] Incorrect x_3 is replaced with x_5.

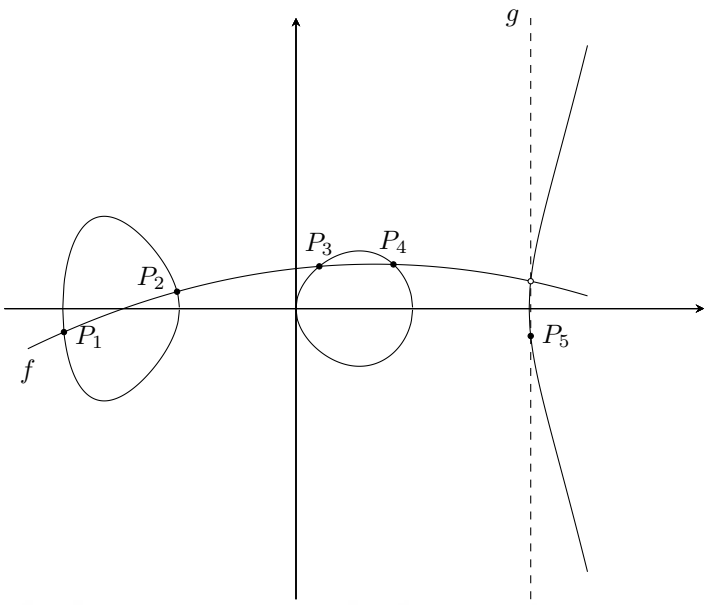

Figure 3.6. $D_1 \sim (P_1) + (P_2) - 2(\mathcal{O}), \quad D_2 \sim (P_3) + (P_4) - 2(\mathcal{O}), \quad C = 0$

∗ Case $C \neq 0$: This is the common case where the function intersects the curve at two more points.

$$
\begin{array}{lcl}
D_1 + D_2 & = & (P_1) + (P_2) + (P_3) + (P_4) - 4(\mathcal{O}) \\
(f) & = & (P_1) + (P_2) + (P_3) + (P_4) + (-P_5) + (-P_6) - 6(\mathcal{O}) \\
- & & \\
\hline
 & = & -(-P_5) - (-P_6) + 2(\mathcal{O}) \\
(g_1) & = & (P_5) + (-P_5) - 2(\mathcal{O}) \\
(g_2) & = & (P_6) + (-P_6) - 2(\mathcal{O}) \\
+ & & \\
\hline
D_1 + D_2 & \sim & (P_5) + (P_6) - 2(\mathcal{O})
\end{array}
$$

See Figure 3.6 for a geometric interpretation of the case.

Here, the common addition formula applies. Furthermore, this is the most frequent case in most cryptographic applications. The formulas are reproduced here to keep the text self-contained; see (Hisil & Costello, 2017, §5, §6) for operation counts and further information.

$$
\begin{aligned}
q_3 &= (q_1 - q_2) + 2\frac{A}{C} - \frac{B^2}{C^2}, \\
r_3 &= (q_1 - q_2)\frac{A}{C} + \frac{A^2}{C^2} + (q_1 + q_2)\frac{B^2}{C^2} - (s_1 + s_2)\frac{B}{C}, \\
s_3 &= (r_1 - r_3)\frac{C}{B} - q_3(q_1 - q_3)\frac{C}{B} + (q_1 - q_3)\frac{A}{B} - s_1, \\
t_3 &= (r_1 - r_3)\frac{A}{B} - r_3(q_1 - q_3)\frac{C}{B} - t_1.
\end{aligned}
\tag{3.9}
$$

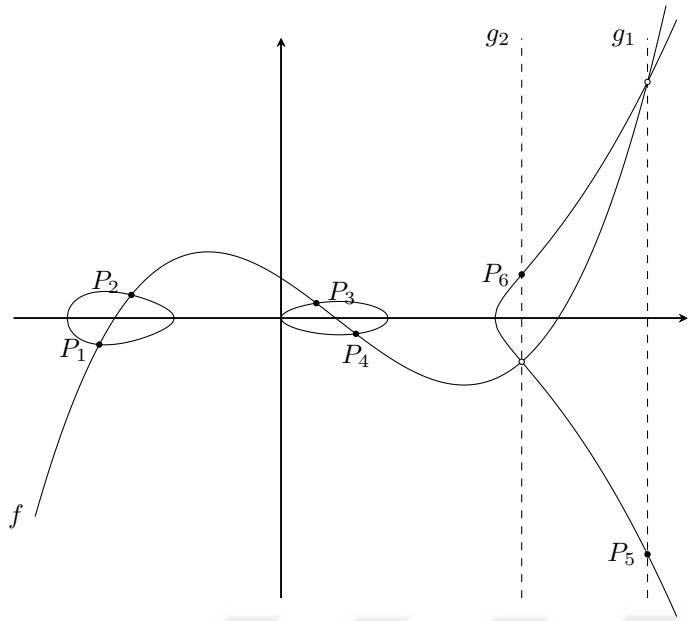

Figure 3.7. $D_1 \sim (P_1) + (P_2) - 2(\mathcal{O}), \quad D_2 \sim (P_3) + (P_4) - 2(\mathcal{O})$

[Script 14] in the Appendix verifies the correctness of this formula.

CHAPTER 4
JACOBIAN COORDINATES

The first inversion-free projective formulas for divisor addition on genus 2 hyperelliptic curves are proposed by Lange (2002c) 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 the equation $(\lambda Q_1 \colon \lambda R_1 \colon \lambda S_1 \colon \lambda T_1 \colon \lambda Z_1) = (q_1 \colon r_1 \colon s_1 \colon t_1 \colon 1)$ 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 $(\lambda^2 Q_1 \colon \lambda^2 R_1 \colon \lambda^3 \mu S_1 \colon \lambda^3 \mu T_1 \colon \lambda Z_1 \colon \mu W_1) = (q_1 \colon r_1 \colon s_1 \colon t_1 \colon 1 \colon 1)$ and $\lambda, \mu \neq 0$ in (Lange, 2002d).

Hisil and Costello (2017) introduced Jacobian coordinates for hyperelliptic curves of genus 2. 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 $(\lambda^2 Q_1 \colon \lambda^4 R_1 \colon \lambda^3 \mu S_1 \colon \lambda^5 \mu T_1 \colon \lambda Z_1 \colon \mu W_1) = (q_1 \colon r_1 \colon s_1 \colon t_1 \colon 1 \colon 1)$ and $\lambda, \mu \neq 0$. Refer to (Hisil & Costello, 2017) for further details.

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

The inversion-free formulas in weighted Jacobian coordinates are given only for the generic addition and doubling (Hisil & Costello, 2017) and for the cases that contain degenerate divisors (Hu et al., 2021). Moreover, the degenerate divisors in (Hu et al., 2021) are considered to be in affine coordinates.

By following the same order as in Section 3, inversion-free formulas with all divisors in weighted Jacobian coordinates are presented here for the sake of a complete inversion-free divisor addition algorithm. The cases already given in (Hisil & Costello, 2017; Hu et al., 2021) are omitted by referring the reader to the mentioned papers. Finally, an optimized implementation of the complete and inversion-free divisor addition algorithm is given at `https://github.com/ozbayelif/jac-on-jac` in the Magma language (Bosma, Cannon, & Playoust, 1997).

Throughout this section, the output divisors is represented as $D_3 = (P_5) - (\mathcal{O}) =$

(X_5, Y_5, Z_5, W_5) if degenerate and $D_3 = (P_5)+(P_6)-2(\mathcal{O}) = (Q_3, R_3, S_3, T_3, Z_3, W_3)$ if non-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 (Hu et al., 2021, §4.4); however, the input divisors are considered to be in affine coordinates.

The formula for the case $D_1 = D_2$ below corresponds to Equation (2.3).

$$\begin{aligned} Q_3 &= -2X_1 \\ R_3 &= X_1^2 \\ S_3 &= \left(a_1 Z_1^8 + 2a_2 X_1 Z_1^6 + 3a_3 X_1^2 Z_1^4 + 5X_1^4\right) W_1 \\ T_3 &= \left(2a_0 Z_1^{10} + a_1 X_1 Z_1^8 - a_3 X_1^3 Z_1^4 - 3X_1^5\right) W_1 \\ Z_3 &= Z_1 \\ W_3 &= 2Y_1 \end{aligned} \tag{4.1}$$

The inversion-free version of Equation (2.2) for $D_1 \neq D_2$ is as follows.

$$\begin{aligned} Q_3 &= -X_1 Z_2^2 - X_2 Z_1^2 \\ R_3 &= X_1 Z_2^2 X_2 Z_1^2 \\ S_3 &= Y_1 Z_2^5 W_2 - Y_2 Z_1^5 W_1 \\ T_3 &= \left(X_1 Y_2 Z_1^3 W_1 - X_2 Y_1 Z_2^3 W_2\right) Z_1^2 Z_2^2 \\ Z_3 &= Z_1 Z_2 \\ W_3 &= \left(X_1 Z_2^2 - X_2 Z_1^2\right) W_1 W_2 \end{aligned} \tag{4.2}$$

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 Section 3.1.3, 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.

$$\begin{aligned} X_3 &= X_1 & X_4 &= -Q_2 Z_1^2 - X_3 Z_2^2 \\ Y_3 &= S_2 Z_1^3 X_3 Z_2^2 + T_2 Z_1^5 & Y_4 &= S_2 Z_1^3 X_4 + T_2 Z_1^5 \\ Z_3 &= Z_1 & Z_4 &= Z_1 Z_2 \\ W_3 &= W_2 Z_2^5 & W_4 &= W_2 \end{aligned} \tag{4.3}$$

The inversion-free tripling formula in (Hu et al., 2021), 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 Equation (3.1).

$$\begin{aligned} F' &= \left(a_1 Z_1^8 + 2a_2 X_1 Z_1^6 + 3a_3 X_1^2 Z_1^4 + 5X_1^4\right) W_1 \\ F'' &= 2\left(a_2 Z_1^6 + 3a_3 X_1 Z_1^4 + 10 X_1^3\right) \\ K &= 8 Y_1^3 \\ A &= \left(2F'' Y_1^2 - F'^2\right) W_1 \\ B &= 2F' Y_1^2 - A X_1 \\ C &= Y_1 K - \left(A X_1 + 2B\right) X_1 W_1 \\ Z_3 &= K Z_1 \\ Q_3 &= 3 X_1 K^2 - A^2 \\ R_3 &= a_3 Z_3^4 - \left(4AB - 3X_1\left(Q_3 - X_1 K^2\right)\right) K^2 \\ S_3 &= A Q_3 W_1 - 2B K^2 W_1 \\ T_3 &= A R_3 W_1 - C K^4 \\ W_3 &= W_1 \end{aligned} \tag{4.4}$$

The following case is where $P_1 = P_3 \neq P_4$. Let $2D_1 = (Q, R, S, T, Z, W)$ be computed with Equation (4.1). Then, $2D_1$ and $(P_4) - (\mathcal{O})$ can be added, as in Equation Equation (3.2), with the below formula.

$$\begin{aligned} K &= \left(Q Z^2 X_4 Z_4^2 + X_4^2 Z^4 + R Z_4^4\right) W_4 \\ A &= Y_4 Z^5 W - \left(S Z_4^3 X_4 Z^2 + T Z_4^5\right) W_4 \\ B &= S K Z_4 + A Q \\ C &= T K Z_4 + A R \\ L &= K Z_4 W \\ M &= K Z W \\ Q_3 &= X_4 M^2 - Q L^2 - A^2 \\ R_3 &= \left(a_3 Z^4 + Q^2 - R\right) L^4 - A\left(B + K S Z_4\right) L^2 + Q_3 X_4 M^2 \end{aligned}$$

$$
\begin{aligned}
S_3 &= AQ_3 - BL^2 \\
T_3 &= AR_3 - CL^4 \\
Z_3 &= KZZ_4W \\
W_3 &= 1
\end{aligned}
\tag{4.5}
$$

The last case corresponds to Equation (3.3), where the supports are disjoint. The inversion-free formula is given below.

$$
\begin{aligned}
K &= \left(Q_2Z_2^2X_1Z_1^2 + X_1^2Z_2^4 + R_2Z_1^4\right)W_1 \\
A &= Y_1Z_2^5W_2 - \left(S_2Z_1^3X_1Z_2^2 + T_2Z_1^5\right)W_1 \\
B &= S_2KZ_1 + AQ_2 \\
C &= T_2KZ_1 + AR_2 \\
L &= KZ_1W_2 \\
M &= KZ_2W_2 \\
Q_3 &= X_1M^2 - Q_2L^2 - A^2 \\
R_3 &= \left(a_3Z_2^4 + Q_2^2 - R_2\right)L^4 - A\left(B + KS_2Z_1\right)L^2 + Q_3X_1M^2 \\
S_3 &= AQ_3 - BL^2 \\
T_3 &= AR_3 - CL^4 \\
Z_3 &= KZ_1Z_2W_2 \\
W_3 &= 1
\end{aligned}
\tag{4.6}
$$

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 $D_2 = (P_3) + (P_4) - 2(\mathcal{O}) = (Q_2, R_2, S_2, T_2, Z_2, W_2)$.

The cases where the output is degenerate are omitted; see (Hu et al., 2021, §4.3). Likewise, refer to (Hisil & Costello, 2017) for the common subexpressions in Jacobian coordinates and the formulas for the common cases. Throughout this section, the points in the supports are notated as $P_i = (X_{P_i}, Y_{P_i}, Z_{P_i}, W_{P_i})$ to avoid confusion between the Z, W coordinates of points and divisors.

Next is the doubling, where $u_1 = u_2$ and $v_1 = v_2$. 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 Equation (4.1).

$$
\begin{aligned}
X_{P_2} &= -T_1 \\
X_{P_1} &= -\left(Q_1S_1 - T_1\right)S_1
\end{aligned}
$$

$$
\begin{aligned}
Y_{P_1} &= \left(S_1 T_1 + X_{P_1}\right) S_1^4 \\
Z_{P_1} &= S_1 Z_1 \\
W_{P_1} &= W_1
\end{aligned} \tag{4.7}
$$

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 Equation (4.1).

$$
\begin{aligned}
K &= S_1 Z_2^3 W_2 - S_2 Z_1^3 W_1 \\
X_{P_1} &= \left(T_2 Z_1^5 W_1 - T_1 Z_2^5 W_2\right) K \\
Y_{P_1} &= \left(S_1 X_{P_1} + T_1 Z_2^2 K^2\right) K^3 Z_2^3 \\
Z_{P_1} &= K Z_1 Z_2 \\
W_{P_1} &= W_1
\end{aligned} \tag{4.8}
$$

Now the addition case where $u_1 \neq u_2$ is investigated. When $B = 0$, the coordinates of the points in the supports can be calculated as below.

$$
\begin{aligned}
K &= Q_1 Z_2^2 - Q_2 Z_1^2 \\
X_{P_1} &= \left(R_2 Z_1^4 - R_1 Z_2^4\right) K \\
Y_{P_1} &= \left(S_1 X_{P_1} + T_1 K^2 Z_2^2\right) K^3 Z_2^3 \\
Z_{P_1} &= Z_1 Z_2 K \\
W_{P_1} &= W_1 \\
\\
X_{P_2} &= X_{P_1} \\
Y_{P_2} &= \left(S_2 X_{P_2} + T_2 K^2 Z_1^2\right) K^3 Z_1^3 \\
Z_{P_2} &= Z_{P_1} \\
W_{P_2} &= W_2 \\
\\
X_{P_3} &= -Q_1 K^2 Z_2^2 - X_{P_1} \\
Y_{P_3} &= \left(S_1 X_{P_3} + T_1 K^2 Z_2^2\right) K^3 Z_2^3 \\
Z_{P_3} &= Z_{P_1} \\
W_{P_3} &= W_1 \\
\\
X_{P_4} &= -Q_2 K^2 Z_1^2 - X_{P_2} \\
Y_{P_4} &= \left(S_2 X_{P_4} + T_2 K^2 Z_1^2\right) K^3 Z_1^3 \\
Z_{P_4} &= Z_{P_1} \\
W_{P_4} &= W_2
\end{aligned} \tag{4.9}
$$

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 Equation (4.2).

If $Y_{P_1} = Y_{P_3}$, the result can be calculated by doubling P_1 with Equation (4.1) and adding P_2 and P_3 with Equation (4.2).

CHAPTER 5
COMPUTATIONAL ASPECTS

In the scope of this work, the formulas given in Chapter 3 and Chapter 4 are generated and verified with the help of the mathematical languages; Magma and Maple ™. The formulas are formed by parametrically running Cantor's algorithm for different cases of divisor addition. Each case of divisor addition occurs due to a condition (i.e., $B = 0$). A quotient ring can be built where these conditions are met, thanks to Magma's built-in algebraic functions. Finally, executing Cantor's algorithm over the quotient ring allows us to produce the formulas for each case. That way, the polynomial arithmetic of Cantor's algorithm is explicated. However, formulas produced in this way are far from their simplified forms. At this point, Maple stepped in with its simplification methods based on elimination theory. By incorporating what is known into the system, simpler forms of the formulas with reduced operation counts are found. Maple also has been used to generate the plots given in the text, which are later drawn in TikZ package in LaTeX. The simplified formulas are then verified in Magma algebraically through quotient rings; see the scripts in the Appendix.

The implementation of the complete divisor addition algorithm can be accessed through the repository link given in the text. The implementation consists of; the sub-functions containing the explicit formulas for each case, a main function that determines the case for the given input and calls the corresponding sub-function, and a test function that adds all divisors of a curve with all other divisors and compares the result with the result of Magma's built-in function. The test function runs repeatedly for different base fields and different genus 2 curves. All of the functions are implemented for both affine and Jacobian coordinates.

The computations were done on a 64-bit Linux environment running on an AMD Ryzen 7 4700U processor. Magma 2.16-9 is used for the implementation of the complete divisor addition algorithm, which is compatible with the latest version available on Web-Magma. Maple 2019 is used for simplifications and the generation of the plots.

CHAPTER 6
CONCLUSION

Elliptic curves were studied in depth from many aspects and have been the basis of several cryptosystems with new developments that are constantly taking place. Besides, hyperelliptic curves have a much richer structure, but many things that are trivial in elliptic curves are not yet applicable to hyperelliptic curves. Rather than pointing out that higher genus hyperelliptics are not viable, it does indicate that much remains to be discovered. Genus 2 curves being faster than elliptic curves in some cases is the strongest indicator of this.

There are sources that introduce the aspects of hyperelliptic curves in a simple and understandable language, see (Menezes et al., 1998; Cohen et al., 2012; Galbraith, 2012). However, the subject of higher genus hyperelliptic curves is usually explained in many books with an advanced language, e.g. (Cassels & Flynn, 1996). Furthermore, the interpretation of the arithmetic on hyperelliptic curves, along with its analogy with elliptic curve arithmetic, is lacking in the literature. This thesis illustrates the group law on genus 2 hyperelliptic curves in terms of geometry, polynomial arithmetic, divisor arithmetic, and explicit formulas. So, one can see the connection between the cubic intersecting the curve, the polynomial representing that cubic in Cantor's algorithm, and the function divisor representing the cubic. This 4-fold interpretation welcomes the readers to the topic as a comprehensive introduction.

One other concern of this work was cryptographic applications. The applicability of hyperelliptic curves in cryptography is directly proportional to the speed of the arithmetic on it. Although it is momentous that Cantor's algorithm enables divisor addition in all genus, the algorithm has no claim in terms of speed. This has led the direction of research to turn to explicit formulas that promise performance. In cryptographic applications, inversion-free formulas in projective coordinates are always preferred to affine formulas. This is simply because the division operation is expensive by means of time complexity. Projective formulas in different weights are given for genus 2 hyperelliptic curves in the literature. The Jacobian coordinates reduced the operation counts significantly but only applied to common addition formulas. Later, Jacobian coordinates are used for cases that include a degenerate divisor, either in the inputs or the output. The other cases, which are not common but also do not incorporate any degenerate divisor, remain to be a gap in the literature.

In this thesis, the complete arithmetic on hyperelliptic curves of genus 2 is examined as a completion of the works in (Lange, 2002c; Hisil & Costello, 2017; Hu et al., 2021). The formulas are derived from Cantor's algorithm and merged in a procedure that is structured in regard to the degrees of the divisors and the intersection of their supports. The algorithm is implemented in Magma language, and an access link to the script is provided within the text. The verifications of the formulas are also provided as Magma scripts. The formulas are given in affine coordinates for all common and rare cases and in inversion-free Jacobian coordinates for the missing cases in the literature. While the weighting of Jacobian coordinates seems natural for the cases with a non-degenerate output, it does not fit that well for the cases with a degenerate output and causes some inefficiencies. On the other hand, the need for a ready-to-use, complete divisor addition algorithm for genus 2 hyperelliptic curves on Jacobian coordinates is met within this thesis. Anyone wishing to build a cryptosystem based on such curves can directly inherit the open-access Magma code through the repository link provided in the text. The formulas given are optimized by eliminating the common subexpressions, and what remains is the migration of the script to a low-level programming language. The parallelization of the algorithm can be considered as future work.

The advancements in generic forms of genus 2 hyperelliptic curves may seem worthless and impractical after the discovery of the Kummer surface. However, many cryptosystems require a prime order genus 2 Jacobian, which is not possible for the Kummer surface. 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.

As a result, hyperelliptic curves have a rich structure waiting to be discovered. It has been seen how fast and competitive operation counts can be reached when the genus is fixed. This thesis aims to pave the way for studies in this field with a guide and a framework for cryptosystems based on generic genus 2 hyperelliptics. The best operation counts are not targeted in the given formulas. However, the proposal of generic and optimized formulas in Jacobian coordinates that handle attacks exploiting the vulnerabilities in degenerate divisors is targeted. In addition, an idea of the overall performance of the Jacobian coordinates is presented.

REFERENCES

Adleman, L. M., DeMarrais, J., & Huang, M.-D. (1994). A subexponential algorithm for discrete logarithms over the rational subgroup of the jacobians of large genus hyperelliptic curves over finite fields. In L. M. Adleman & M.-D. Huang (Eds.), *Algorithmic number theory* (pp. 28–40). Berlin, Heidelberg: Springer Berlin Heidelberg.

Avanzi, R. M. (2004). Aspects of hyperelliptic curves over large prime fields in software implementations. In M. Joye & J.-J. Quisquater (Eds.), *Cryptographic hardware and embedded systems - ches 2004* (pp. 148–162). Berlin, Heidelberg: Springer Berlin Heidelberg.

Barreto, P. S., Galbraith, S. D., hÉigeartaigh, C. Ó., & Scott, M. (2007). Efficient pairing computation on supersingular abelian varieties. *Designs, Codes and Cryptography*, *42*, 239–271.

Bernstein, D. J., Chuengsatiansup, C., Lange, T., & Schwabe, P. (2014). Kummer strikes back: New DH speed records. In P. Sarkar & T. Iwata (Eds.), *Advances in cryptology – asiacrypt 2014* (pp. 317–337). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-662-45611-8_17

Bosma, W., Cannon, J., & Playoust, C. (1997). The Magma algebra system. I. The user language. *J. Symbolic Comput.*, *24*(3-4), 235–265. Retrieved from http://dx.doi.org/10.1006/jsco.1996.0125 (Computational algebra and number theory (London, 1993)) doi: 10.1006/jsco.1996.0125

Cantor, D. G. (1987). Computing in the jacobian of a hyperelliptic curve. *Mathematics of Computation*, *48*(177), 95–101. Retrieved 2023-02-14, from http://www.jstor.org/stable/2007876 doi: 10.2307/2007876

Cassels, J. W. S., & Flynn, E. V. (1996). *Prolegomena to a middlebrow arithmetic of curves of genus 2* (Vol. 230). Cambridge University Press.

Cohen, H., Frey, G., Avanzi, R., Doche, C., Lange, T., Nguyen, K., & Vercauteren, F. (2012). *Handbook of elliptic and hyperelliptic curve cryptography, second edition* (2nd ed.). Chapman & Hall/CRC.

Costello, C., & Lauter, K. (2012). Group law computations on jacobians of hyperelliptic curves. In A. Miri & S. Vaudenay (Eds.), *Selected areas in cryptography* (pp. 92–117). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-642-28496-0_6

Costello, C., & Stebila, D. (2010, 08). Fixed argument pairings. In (Vol. 2010, p. 342).

doi: 10.1007/978-3-642-14712-8_6

Daniel, S. (1971). Class number, a theory of factorization, and genera. In *Proc. sympos. pure math.* (Vol. 20, pp. 415–440).

Diffie, W., & Hellman, M. E. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, *22*(6), 644–654.

Duursma, I., & Lee, H.-S. (2003). Tate pairing implementation for hyperelliptic curves y2 = xp-x + d. In *Asiacrypt.*

Enge, A. (2002). Computing discrete logarithms in high-genus hyperelliptic jacobians in provably subexponential time. *Mathematics of computation*, *71*(238), 729–742.

Flassenberg, R., & Paulus, S. (1999). Sieving in function fields. *Experimental Mathematics*, *8*(4), 339–349.

Galbraith, S. D. (2012). *Mathematics of public key cryptography*. Cambridge University Press.

Gaudry, P. (2000). An algorithm for solving the discrete log problem on hyperelliptic curves. In B. Preneel (Ed.), *Advances in cryptology — eurocrypt 2000* (pp. 19–34). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/3-540-45539-6_2

Gaudry, P. (2007). Fast genus 2 arithmetic based on theta functions. *Journal of Mathematical Cryptology*, *1*(3), 243–265. doi: 10.1515/JMC.2007.012

Gaudry, P., & Harley, R. (2000). Counting points on hyperelliptic curves over finite fields. In W. Bosma (Ed.), *Algorithmic number theory* (pp. 313–332). Berlin, Heidelberg: Springer Berlin Heidelberg.

Gaudry, P., Hess, F., & Smart, N. (2002). Constructive and destructive facets of Weil descent on elliptic curves. *Journal of Cryptology*, *15*, 19-46. Retrieved from https://inria.hal.science/inria-00512763 doi: 10.1007/s00145-001-0011-x

Gaudry, P., Thomé, E., Thériault, N., & Diem, C. (2007). A double large prime variation for small genus hyperelliptic index calculus. *Mathematics of Computation*, *76*(257), 475–492. Retrieved 2023-05-13, from http://www.jstor.org/stable/40234388

Hess, F., Seroussi, G., & Smart, N. P. (2001). Two topics in hyperelliptic cryptography. In S. Vaudenay & A. M. Youssef (Eds.), *Selected areas in cryptography* (pp. 181–189). Berlin, Heidelberg: Springer Berlin Heidelberg.

Hisil, H., & Costello, C. (2017). Jacobian coordinates on genus 2 curves. *Journal of Cryptology*, *30*, 572–600. doi: 10.1007/s00145-016-9227-7

Hu, Z., Lin, D., & Zhao, C.-A. (2021). Fast scalar multiplication of degenerate divisors for hyperelliptic curve cryptosystems. *Applied Mathematics and Computation*, *404*, 126239. Retrieved from https://www.sciencedirect.com/science/article/pii/S0096300321003295 doi: 10.1016/j.amc.2021.126239

Katagi, M., Akishita, T., Kitamura, I., & Takagi, T. (2005). Some improved algorithms for hyperelliptic curve cryptosystems using degenerate divisors. In C.-s. Park & S. Chee (Eds.), *Information security and cryptology – icisc 2004* (pp. 296–312). Berlin, Heidelberg: Springer Berlin Heidelberg.

Katagi, M., Kitamura, I., Akishita, T., & Takagi, T. (2003). *Novel efficient implementations of hyperelliptic curve cryptosystems using degenerate divisors.* Retrieved from http://eprint.iacr.org/2003/203 (This is a full version of WISA 2004 paper. Masanobu.Katagi@jp.sony.com 12643 received 26 Sep 2003, last revised 12 Aug 2004)

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of Computation*, *48*(177), 203–209.

Koblitz, N. (1989, jan). Hyperelliptic cryptosystems. *Journal of Cryptology*, *1*(3), 139-–150. Retrieved from https://doi.org/10.1007/BF02252872 doi: 10.1007/BF02252872

Koblitz, N., Menezes, A. J., Wu, Y.-H., & Zuccherato, R. J. (1998). *Algebraic aspects of cryptography*. Berlin, Heidelberg: Springer-Verlag. doi: 10.1007/978-3-662-03642-6

Lang, S. (1959). Introduction to algebraic geometry. *Bull. Amer. Math. Soc*, *65*, 341–342.

Lange, T. (2002a). *Efficient arithmetic on genus 2 hyperelliptic curves over finite fields via explicit formulae.* Cryptology ePrint Archive, Paper 2002/121. Retrieved from https://eprint.iacr.org/2002/121

Lange, T. (2002b). *Efficient arithmetic on hyperelliptic curves.* Cryptology ePrint Archive, Report 2002/107. (https://ia.cr/2002/107)

Lange, T. (2002c). *Inversion-free arithmetic on genus 2 hyperelliptic curves.* Cryptology ePrint Archive, Report 2002/147. Retrieved from https://eprint.iacr.org/2002/147

Lange, T. (2002d). *Weighted coordinates on genus 2 hyperelliptic curves.* Cryptology ePrint Archive, Report 2002/153. Retrieved from https://eprint.iacr.org/2002/153

Lenstra Jr, H. W., Pila, J., & Pomerance, C. (1993). A hyperelliptic smoothness test. i. *Philosophical Transactions of the Royal Society of London. Series A: Physical*

and Engineering Sciences, *345*(1676), 397–408.

Matsuo, K., Chao, J., & Tsujii, S. (2001, 07). Fast genus two hyperelliptic curve cryptosystems. , *2001*(75), 89–96. Retrieved from https://cir.nii.ac.jp/crid/1520853834452020224

Menezes, A., Zuccherato, R., & Wu, Y.-H. (1998). An elementary introduction to hyperelliptic curves. In N. Koblitz (Ed.), *Algebraic aspects of cryptography* (pp. 155–178). Berlin, Heidelberg: Springer-Verlag.

Miller, V. S. (1986). Use of elliptic curves in cryptography. In H. C. Williams (Ed.), *Advances in cryptology — crypto '85 proceedings* (pp. 417–426). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/3-540-39799-X_31

Miyamoto, Y. (2002). A fast addition algorithm of genus two hyperelliptic curves. *Proc. SCIS2002*.

Montgomery, P. L. (1987). Speeding the pollard and elliptic curve methods of factorization. *Mathematics of computation*, *48*(177), 243–264.

Müller, V., Stein, A., & Thiel, C. (1999). Computing discrete logarithms in real quadratic congruence function fields of large genus. *Mathematics of Computation*, *68*(226), 807–822.

Mumford, D. (1984). Tata lectures on theta II. In *Progress in mathematics* (Vol. 43). Boston, MA: Birkhauser. doi: 10.1007/978-0-8176-4578-6

Nagao, K. (2000). Improving group law algorithms for jacobians of hyperelliptic curves. In W. Bosma (Ed.), *Ants 2000* (Vol. 1838, pp. 439–448). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/10722028_28

Pollard, J. M. (1978). Monte carlo methods for index computation (mod p). *Mathematics of computation*, *32*(143), 918–924.

Silverman, J. H. (1986). *The arithmetic of elliptic curves* (1st ed.). Springer New York. doi: 10.1007/978-1-4757-1920-8

Silverman, J. H. (1994). *Advanced topics in the arithmetic of elliptic curves* (Vol. 151). Springer Science & Business Media.

Smart, N. (1999). On the performance of hyperelliptic cryptosystems. In J. Stern (Ed.), *Advances in cryptology - eurocrypt '99* (Vol. 1592, p. 165-175). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/3-540-48910-X_12

Spallek, A. (1994). *Kurven vom geschlecht 2 und ihre anwendung in public-key-kryptosystemen* (Doctoral dissertation, University of Duisburg-Essen, Germany).

Retrieved from https://d-nb.info/943571049

Stahlke, C. (2004). Point compression on jacobians of hyperelliptic curves over $\mathbb{F}_q$. *Cryptology ePrint Archive*.

Sugizaki, H., Matsuo, K., Chao, J., & Tsujii, S. (2002). An extension of harley addition algorithm for hyperelliptic curves over finite fields of characteristic two. *IEICE Technical Report*.

Takahashi, M. (2002). Improving Harley algorithms for jacobians of genus 2 hyperelliptic curves. *Proc. of SCIS2002*, 155–160.

Thériault, N. (2003). Index calculus attack for hyperelliptic curves of small genus. In C.-S. Laih (Ed.), *Advances in cryptology - asiacrypt 2003* (pp. 75–92). Berlin, Heidelberg: Springer Berlin Heidelberg.

Wollinger, T., Pelzl, J., & Paar, C. (2005). Cantor versus harley: optimization and analysis of explicit formulae for hyperelliptic curve cryptosystems. *IEEE Transactions on Computers*, *54*(7), 861-872. doi: 10.1109/TC.2005.109

Wollinger, T. J. (2004). *Software and hardware implementation of hyperelliptic curve cryptosystems* (Unpublished doctoral dissertation). Ruhr University Bochum.

APPENDIX A
VERIFICATION SCRIPTS

This section contains Magma (Bosma et al., 1997) scripts that verify the formulas presented in the text and assists in proving selected lemmas. These scripts can be executed freely online through Web-Magma (http://magma.maths.usyd.edu.au/calc/), noting that each script requires [Script 20] to be loaded in advance.

```
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;
```

Code A.1. Cantor's Algorithm for genus 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;
```

Code A.2. Verification of Equation (2.3) and Section 3.1.2 Case "$y_1 = y_2$"

```
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;
```

Code A.3. Verification of Equation (2.2) and Section 3.1.2 Case "$x_1 \neq x_2$"

```
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;
```

Code A.4. Verification of Equation (3.1)

```
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;
```

Code A.5. Verification of Equation (3.2)

```
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;
```

Code A.6. Verification of Equation (3.3)

```
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;
```

Code A.7. Verification of Equation (3.5)

```
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;
```

Code A.8. Verification of Equation (3.6)

```
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;
```

Code A.9. Verification of Equation (3.8)

```
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;
```

Code A.10. Verification of Equation (3.9)