UNIVERSITY OF CALGARY

Index Calculus in the Infrastructure of Real Quadratic Function Fields

by

Jonathan Francis Hammell

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

July, 2008

UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Index Calculus in the Infrastructure of Real Quadratic Function Fields" submitted by Jonathan Francis Hammell in partial fulfilment of the requirements for the degree of Master of Science.

Supervisor, Dr. Michael J. Jacobson, Jr.
Department of Computer Science
University of Calgary

Dr. Renate Scheidler
Department of Computer Science
University of Calgary

Dr. Laurent Imbert
Centre National de la Recherche Scientifique
Montpellier, France

July 21, 08
Date

Abstract

Quadratic function fields were first extensively studied by Artin in 1921. These function fields have geometric properties related to hyperelliptic curves as well as number-theoretic properties related to quadratic number fields. In 1972, Shanks introduced the infrastructure of a real quadratic number field. In this work we study the infrastructure of a real quadratic function field.

We provide a heuristic analysis of a new, practical method for performing index calculus in the infrastructure when the genus of the function field is large. We implemented this method, as well as a sieve-based method, and this thesis provides experimental results for computing the regulator that compares the two variants. This is the first known implementation of index calculus in the infrastructure.

Acknowledgements

First and foremost, I would like to thank my supervisor Dr. Michael Jacobson for giving me this project and his guidance throughout its progress. He was always available for questions which was particularly appreciated. I would also like to thank my committee members, Dr. Renate Scheidler and Dr. Laurent Imbert, for their careful reading and critique of my thesis. Thank you to Prof. Hugh Williams for his generous financial assistance under the iCORE Chair for Algorithmic Number Theory and Cryptography. For additional financial support I also thank the Natural Sciences and Engineering Research Council of Canada (NSERC), Alberta's Informatics Circle of Research Excellence (iCORE), Alberta Scholarship Programs, and the Department of Computer Science at the University of Calgary.

I would like to thank my officemates, in particular Karel Bergmann, Mark Velichka and Ryan Vogt, for great discussions and their help and interest in my work. I especially thank Mark Velichka for answering my numerous questions and his assistance with the codebase. Of course, many others provided helpful discussions during my thesis work and for that I would like to thank all of those from the math department in the Centre for Information Security and Cryptography (CISaC) at the University of Calgary.

Thank you to my family and friends for their support and encouragement. Finally, a very special thank you goes to Melissa for her understanding and utmost patience.

Table of Contents

List of Theorems

List of Algorithms

List of Tables

List of Figures

List of Symbols

(x, y)	Point in $\mathbb{A}^2$
$(x : y : z)$	Point in homogeneous coordinates in $\mathbb{P}^2$
(a, b)	Standard representation of a primitive $\mathcal{O}$-ideal (p. 49)
(s, a, b)	Standard representation of a fractional $\mathcal{O}$-ideal (p. 49)
$[L : K]$	Degree of a field extension L/K
$[\alpha_0, \alpha_1, \ldots]$	Continued fraction expansion (p. 58)
$\lceil a \rceil$	Ceiling operation, *i.e.* the smallest integer greater than or equal to a
$\lfloor a \rfloor$	Floor operation, *i.e.* the largest integer less than or equal to a
$\wedge$	Logicial AND
$\vee$	Logical OR
$\approx$	Approximately equal
$\lessapprox$	Less than, but almost equal
$\sim$	Equivalence relation
$\cong$	Isomorphic
$\leftarrow$	Assignment
$\triangleright$	Algorithm comment
$\oplus$	Direct sum operator

$*$	Giant step operation on ideals in the infrastructure (p. 68)
$\circledast$	'Corrected' giant step operation on ideals in the infrastructure (p. 71)
$**$	Perform a scalar multiple corrected giant step operations (p. 77)
$A_{\mathcal{R}}$	Matrix of relations from the set $\mathcal{R}$ (p. 105)
$A'_{\mathcal{R}}$	Matrix of relations augmented with the distances from the set $\mathcal{R}$ (p. 113)
$\mathbb{A}^n$	n-dimensional affine space
$\mathcal{A}_F$	Adèle space of a function field F (p. 36)
$\overline{\alpha}$	Conjugate of an element $\alpha \in F$ (p. 53)
$\overline{\mathfrak{a}}$	Conjugate of an ideal $\mathfrak{a}$ (p. 54)
B	Smoothness bound for a factor base (p. 89)
B_L	Large prime bound for almost B-smooth ideals (p. 143)
$\mathbb{C}$	Field of complex numbers
$C(K)$	Set of K-rational points on the curve C (p. 13)
$\mathrm{char}(K)$	Characteristic of the field K
$\mathrm{Cl}(F)$	Divisor class group of a function field F (p. 45)
$\mathrm{Cl}(\mathcal{O})$	Ideal class group of a quadratic order $\mathcal{O}$ (p. 54)
$\mathrm{Cl}^0(F)$	Degree-zero divisor class group of a function field F (p. 45)
$\mathrm{Con}_\phi(D)$	Conorm of a divisor D with respect to a morphism ϕ (p. 18)
$\mathrm{Con}_{F'/F}(D)$	Conorm of a divisor D with respect to a function field extension F'/F (p. 35)
$\delta(\mathfrak{a})$	Distance of principal $\mathcal{O}$-ideal $\mathfrak{a}$ from $\mathcal{O}$ (p. 64)
$\delta(\mathfrak{b}, \mathfrak{a})$	Distance of $\mathfrak{b}$ from an equivalent ideal $\mathfrak{a}$ (p. 64)
$d(\mathscr{P}'\|\mathscr{P})$	Different exponent of a place $\mathscr{P}'$ lying over $\mathscr{P}$ (p. 38)

$\deg f$	Degree of a polynomial f, *i.e.* the highest power of the indeterminant with non-zero coefficient
$\deg \mathfrak{a}$	Degree of a reduced ideal $\mathfrak{a}$ (p. 95)
$\det(X)$	Determinant of a matrix X
$\mathrm{Diff}(F'/F)$	Different of a function field extension F'/F (p. 38)
$\mathrm{Div}(C)$	Set of divisors on a curve C (p. 17)
$\mathrm{Div}(F)$	Set of divisors of a function field F (p. 34)
$\mathrm{Div}^d(F)$	Group of divisors of degree d of a function field F (p. 45)
$\mathrm{div}(f)$	Divisor of a function f (pp. 17, 34)
$\mathrm{div}(\omega)$	Canonical divisor for a differential ω (pp. 20, 37)
$\mathrm{div}[a, b]$	Mumford representation of a divisor (p. 48)
$e(P)$	Ramification index of a morphism at a point P (p. 16)
$e(\mathscr{P}'\|\mathscr{P})$	Ramification index of a place $\mathscr{P}'$ lying over $\mathscr{P}$ (p. 33)
$\exp(n)$	Exponential function, *i.e.* $\exp(n) = e^n$ for the natural constant e
f	Polynomial in $K[x]$ from the curve equation $C : y^2 + h = f$ (pp. 25, 43)
$\mathbb{F}_q$	Finite field of order q
$\mathcal{F}_B$	Factor base with smoothness bound B (p. 89)
$\mathrm{Frac}(\mathcal{O})$	Group of fractional $\mathcal{O}$-ideals (p. 54)
g	Genus of a curve or function field (pp. 21, 37)
$g(u)$	Sieve polynomial (p. 132)
$\gcd(a, b)$	Greatest common divisor of a and b
$\mathrm{GL}_n(R)$	General linear group of degree n for a ring R (p. 49)
H_∞	Hyperplane at infinity (p. 13)
h	Polynomial in $K[x]$ from the curve equation $C : y^2 + h = f$ (pp. 25, 43)

h_F	Divisor class number of a function field F (p. 46)
$h_{\mathcal{O}}$	Ideal class number of a quadratic order $\mathcal{O}$ (p. 55)
ι	Hyperelliptic involution (pp. 26, 29,44)
$\overline{K}$	Algebraic closure of the field K
$\widetilde{K}$	Field of constants of an algebraic function field F/K (p. 31)
K^*	Set of units in the field K
$K[C]$	Coordinate ring of an algebraic curve C (p. 13)
$K(C)$	Function field of an algebraic curve C (p. 13)
$K[x]$	Ring of polynomials in the indeterminant x with coefficients in K
$K(x)$	Rational function field in x over K (p. 30)
$K((1/x))$	Field of power series in the variable $1/x$ (pp. 57, 57)
Λ_B	Lattice of relations correponding to a smoothness bound B (p. 112)
Λ'_B	Lattice of relations augmented with distances correponding to a smoothness bound B (p. 112)
λ	Function to determine the next index in the sieve array (p. 133)
$L(n)$	Short form for $\log n \log\log n$ (p. 52)
$\mathscr{L}(D)$	Riemann-Roch space of a divisor D (pp. 20, 37)
$\mathbf{L}_n(c)$	Subexponential function with $\alpha = 1/2$ (p. 88)
$\mathbf{L}_n(\alpha, c)$	Subexponential function (p. 87)
$\ell(D)$	Dimension of the Riemann-Roch space $\mathscr{L}(D)$ (pp. 20, 37)
$\log(a)$	Logarithm of a (to the base 2, unless otherwise specified)
$\mathfrak{m}_{\mathscr{P}}$	Maximal ideal of the valuation ring $\mathcal{O}_{\mathscr{P}}$ (p. 31)
$\mathfrak{m}_{\mathcal{P}}$	Maximal ideal of $\mathcal{O}_{\mathcal{P}}$
$\max S$	Maximum element of a set S

$\min S$	Minimum element of a set S
mod	Modulo reduction (*i.e.* remainder after division)
ν	Map into the sieving array (p. 132)
$\mathbb{N}$	Set of natural numbers (not including zero)
$\mathbb{N}_0$	Set of natural numbers including zero
$\mathrm{N}(\alpha)$	Norm of a function field element α (p. 60)
$\mathrm{N}(\mathfrak{a})$	Norm of an ideal $\mathfrak{a}$ (p. 53)
n_B	Cardinality of a factor base with smoothness bound B (p. 89)
n_G	Order of a group G (p. 89)
$n_{G/B}$	Number of B-smooth elements in G (p. 91)
$\Omega_{F/K}$	Module of differentials of a function field F/K (pp. 19, 36)
ω	Differential (pp. 19, 36)
$\mathcal{O}$	Quadratic order of a function field (p. 48)
$\mathcal{O}_P$	Local ring of a point P (p. 13)
$\mathcal{O}_{\mathscr{P}}$	Valuation ring of a place $\mathscr{P}$ (p. 31)
$\mathcal{O}_{\mathcal{P}}$	Local ring/valuation ring of a prime divisor $\mathcal{P}$
$O(n)$	Big-O notation
$\widetilde{O}(n)$	"Soft" big-O notation, *i.e.* ignoring factors logarithmic with respect to the input size
$o(n)$	Little-o notation
Φ	Frobenius endomorphism or the Frobenius map (p. 102)
P	Point on a curve
$\mathscr{P}$	Place of a function field (p. 31)
$\mathcal{P}$	Prime divisor (p. 46)

P_∞	Point at infinity (p. 13)
$\mathscr{P}_\infty$	Place at infinity (p. 32)
$\mathcal{P}_\infty$	Prime divisor at infinity (p. 47)
$\mathcal{P}_{B_L}$	Set of partial relations corresponding to the large prime bound B_L (p. 143)
$\mathbb{P}^n$	n-dimensional projective space
$\mathrm{Prin}(F)$	Group of principal divisors of the function field F (p. 45)
$\mathrm{Prin}(\mathcal{O})$	Group of principal fractional $\mathcal{O}$-ideals (p. 54)
q	Order of the underlying field
$\mathbb{Q}$	Field of rational numbers
$\mathrm{Quot}(R)$	Field of quotients of a ring R
$\rho(\mathfrak{a})$	Baby step algorithm performed on ideal $\mathfrak{a}$ (p. 62)
$\rho(\mathfrak{a}, \delta(\mathfrak{a}))$	Baby step algorithm performed on ideal $\mathfrak{a}$ returning the updated distance (p. 64)
$\rho^*(\mathfrak{a}, d)$	Find the ideal with distance from $\mathfrak{a}$ closest to d (p. 66)
R_φ	Ramification divisor of a morphism φ (p. 22)
$R_\mathcal{O}$	Regulator of a quadratic order $\mathcal{O}$ (p. 55)
$\mathbb{R}$	Field of real numbers
$\mathbb{R}_{\geq 0}$	Subset of $\mathbb{R}$ consisting of elements greater or equal to 0
$\mathcal{R}_B$	Set of relations corresponding to a factor base with smoothness bound B (p. 105)
$\Sigma(F)$	Set of places of the function field F (p. 31)
σ	Smoothness test algorithm (pp. 103, 144)
$\mathcal{S}_{g,\mathfrak{p}}$	Roots of the sieve polynomial $g(u)$ modulo $\mathrm{N}(\mathfrak{p})$ (p. 133)
$\mathrm{sgn}(f)$	Leading coefficient of a polynomial f

$\mathrm{supp}(D)$	Support of a divisor D (pp. 16, 34)
$\Theta(n)$	Big-Theta notation
ϑ	A constant used to ensure subexponentiality of the factor base (p. 99)
θ_i	The inverse product of complete quotents of a continued fraction expansion (p. 60)
$\mathbf{x}$	Vector
$\mathbf{x}^{\mathrm{tr}}$	Transpose of a vector $\mathbf{v}$
$\mathrm{xgcd}(a, b)$	Extended Euclidean algorithm for computing $d = \gcd(a, b)$ and u, v such that $d = ua + vb$
$\mathbb{Z}$	Set of integers (an integral domain)

The magic words are squeamish ossifrage

(Cryptanalytic result of a ciphertext challenge encoded with RSA-129)

Encoded by Ron Rivest, Adi Shamir and Leonard Adleman, in "Mathematical Games," *Scientific American*, August 1977; Plaintext found by Derek Atkins, Michael Graff, Arjen Lenstra and Paul Leyland, in *Advances in Cryptology—ASIACRYPT'94*, 263–277, Springer-Verlag, 1995

Chapter 1

Introduction

The advent of computers gave rise to a new discipline worked on by both mathematicians and computer scientists called **computational number theory**. It is the study of algorithms for solving number theoretic problems. Computers and efficient algorithms have greatly increased the range that computational number theorists have been able to test theoretical conjectures. Interest in these problems grew in the mid-1970s when it was discovered that the privacy and authenticity of electronic communications could be secured if based on a computational problem that is sufficiently difficult.

1.1 Public Key Cryptography

Public key cryptography was introduced independently by Diffie and Hellman [DH76] and Merkle [Mer78]. The authors of the former paper described how cryptographic key distribution could be simplified if one used a so-called **one-way trapdoor**

function.[1] Such a function is easy to compute, yet computationally difficult to invert without the knowledge of some key piece of information, called the **trapdoor information.** Diffie and Hellman also claimed that this one-way trapdoor function could be used to prove the authenticity of electronic data using a **digital signature.** The security of these cryptographic protocols relies on the computational difficulty required to invert the trapdoor function without *a priori* knowledge of the trapdoor information. Public key cryptography is now used to secure government communications and financial transactions in business, hence there is practical and economic importance to computational number theory.

One of the most famous computational problems in this area is integer factorization. The first example of public key cryptography had a trapdoor function based on integer factorization. Introduced in 1978 by Rivest, Shamir and Adleman [RSA78], this scheme became pervasive in cryptographic applications and is generally referred to as **RSA.**[2] The fastest known algorithms developed by computational number theorists for performing integer factorization are based on a technique called **index calculus.** Although index calculus does not *break* RSA, it significantly weakens it, requiring one to use key sizes of at least 1024 bits versus the 665 bits suggested in the original paper.

In 1984, another public key cryptographic scheme was proposed with a trapdoor function based on the computational problem of finding an integer exponent x given a group element $h = g^x$ for a known group generator g. This problem is called

[1] In 1997 it was revealed in a formally classified document [Ell87] that James Ellis and Malcolm Williamson of the CESG in Great Britain had discovered public key cryptography and the Diffie-Hellman scheme respectively in 1970 and 1974 [Ell70],[Wil74].

[2] It was also revealed in 1997 that Clifford Cocks of the CESG had discovered the RSA scheme in 1973 [Coc73].

the **discrete logarithm problem** (DLP) and the cryptographic scheme based on it was proposed by ElGamal [ElG85]. For certain groups the DLP is computationally difficult if the size of the group is chosen to be large enough. Again, index calculus can be applied to solve an instance of the DLP in many groups, thereby requiring the group size to be increased for the DLP to remain computationally infeasible. However, index calculus is not a so-called **generic method** for solving the DLP since it cannot be applied in every type of group. For instance, computational number theorists have not been able to apply index calculus successfully to the group of points on an elliptic curve. There have been proposals by Silverman [Sil00] and Semaev [Sem04], but the algorithms are less efficient than the generic method of Pollard [Pol78] which runs in square-root time with respect to the order of the group.[3] Consequently, cryptosystems based on the elliptic curve DLP are able to use key sizes that are significantly smaller than RSA and ElGamal in other groups. This advantage allows cryptography to be deployed in a wider range of devices. Therefore, cryptographers have searched for other groups that offer a DLP that is secure against index calculus. Class groups in number fields and function fields showed promise.

1.2 Number Fields and Function Fields

In 1801 Gauss published a list of **imaginary quadratic number fields** and conjectured that this list was complete for class numbers of one, two and three [Gau86, §V, Art. 303]. Computing the class number of a quadratic number field is computationally difficult—Gauss's technique was fully exponential—and it was not until

[3]Index calculus attacks do greatly reduce the security of the elliptic curve DLP over certain types of fields, but these are easy to avoid in practice (*cf.* Gaudry [Gau07] and Diem [Die08, pp. 157–208]).

1985 that Gauss's claim was proven to be correct (*cf.* Goldfeld [Gol85]). Most classical algorithms for computing the class number were based on the Dirichlet L-series. Shanks proposed a method that enables one to compute the class number and class group structure much faster than the classical techniques [Sha71]. Shanks's method, now known as the **baby-step giant-step algorithm**, has been adapted to many other computation problems; however, it is still only a square-root time algorithm.

In 1989, Hafner and McCurley described how to adapt index calculus to the class group of an imaginary quadratic number field and showed how one could compute the class number and the group structure [HM89]. Their method relied upon on the unproven, but widely believed, Extended Riemann Hypothesis (ERH). A year earlier, Buchmann and Williams had proposed a DLP in the class group of an imaginary quadratic number field as the basis for a public key cryptosystem [BW88]. However, it was immediately realized by McCurley that their index calculus method also enables one to solve instances of the DLP [McC89, §5]. More recently, Vollmer presented another technique for index calculus in quadratic number fields that improves on the Hafner-McCurley method [Vol00].

The case of **real quadratic number fields** is quite different. Cohen and Lenstra developed heuristics that suggest that the ideal class numbers of real quadratic number fields tend to be small [CL84]. Gauss's conjecture that there are infinitely many real quadratic number fields with class number one is still an open problem [Gau86, §V, Art. 304]. Due to the small class numbers, the DLP defined in the imaginary case does not translate directly to real quadratic number fields. However, Shanks recognized an internal structure to the ideal classes, which he called the **infrastructure** [Sha72b]. Buchmann and Williams realized that the infrastructure allowed one

to define a DLP for a real quadratic number field [BW90]. This was the first time the DLP had been extended to a mathematical structure that is not a group. The scheme was improved upon by Scheidler, Buchmann and Williams [SBW91, Sch93] and, more recently, by Jacobson, Scheidler and Williams [JSW06]. However, Buchmann also showed that index calculus could be adapted to real quadratic number fields, where in addition to solving the DLP it could be used to find the class number, group structure and the regulator [Buc90]. It is important to note that the index calculus algorithms of Hafner-McCurley and Buchmann were verified in practice through implementations by Buchmann and Düllmann [BD91, BD92] and Cohen, Diaz y Diaz and Oliver [CDO93, CDO97].

As many algebraic number theorists know, number fields have many similarities with another class of global fields called function fields (over finite fields). Quadratic function fields can be derived using purely algebraic methods or from algebraic geometric structures called hyperelliptic curves. Class groups in **imaginary quadratic function fields** can be directly represented by the Jacobian of a hyperelliptic curve. Schoof [Sch85, Sch95] and Satoh [Sat00] discovered polynomial-time algorithms for computing the class number of a genus-one quadratic function field where the class number corresponds to the number of points on an elliptic curve. Schoof's algorithm has been generalized to to higher genus by Pila [Pil90], Huang and Ierardi [HI98], and Gaudry and Harley [GH00], but their algorithms are only polynomial-time when the genus is fixed, thus only potentially feasible for small genera. Mestre proposed an algorithm related to Satoh's method to count points on the Jacobian for arbitrary genus, but it is still restricted to small genera and small field characteristics [Mes00, Mes02]. A cohomological point counting algorithm introduced by Kedlaya is

polynomial-time in the genus, but exponential in the size of the field characteristic, even with recent improvements by Harvey [Ked01, Ked04], [Har07].

A DLP in the class group of an imaginary quadratic function field was proposed for cryptography by Koblitz [Kob89]. Adleman, DeMarrais and Huang [ADH94] were the first to describe how to adapt index calculus to this DLP with generalizations by Bauer [Bau99, Bau01] and Enge [Eng00, Eng02]. However, the ADH algorithm was predicted to only be efficient when the genus of the function field was large. Gaudry presented another index calculus variant for solving the DLP when the class number is known in advance [Gau00a]. While this does not help the computational problem of finding the class number, it is argued that this value is likely to be known in cryptographic situations. Moreover, Gaudry's algorithm also applies to the small genus case, reducing the space where the DLP still seems to be immune to index calculus to genera less than four. Thériault [Thé03] and Gaudry, Thomé, Thériault and Diem [GTTD07] have described further theoretical improvements to Gaudry's original algorithm.

The first implementations of index calculus in imaginary quadratic function fields were by Paulus and Flassenberg using practical improvements to the ADH algorithm [Pau96, FP99]. Other known implementations targeting the large genus case were by Smart [Sma97], Jacobson, Menezes and Stein [JMS01], and recently by Velichka [Vel08]. Implementations of index calculus for small genus imaginary quadratic function fields have been described by Gaudry [Gau00a] and with the latest improvements by Gaudry, Thomé, Thériault and Diem [GTTD07].

Real quadratic function fields have similar properties as real quadratic number fields. The ideal class number tends to be small, and ideal classes have an internal

structure referred to as the infrastructure. Enge described how index calculus could be applied in the Jacobian of a real hyperelliptic curve [Eng02]. However, since the Jacobian is not isomorphic to the ideal class group of a real quadratic function field, Enge's method does not find the ideal class number, regulator, or the ideal class group structure. Similarly, the methods described above based on point counting for computing the class number of an imaginary quadratic function field would not give the ideal class number of a real quadratic function field without some method for computing the regulator.

Scheidler, Stein and Williams proposed a DLP in the infrastructure of a real quadratic function field [SSW96]. Recently, more cryptographic schemes have been proposed based on the infrastructure DLP by Jacobson, Scheidler and Stein [JSS07a]. Müller, Stein and Thiel described how index calculus could be performed in the infrastructure of a real quadratic function field when the genus is large [MST99]. They described how their algorithm could be used to solve the infrastructure DLP, as well to find the ideal class number, the regulator, and the ideal class group structure. However, their algorithm had not been verified in practice since there was no known implementation until now.

One may ask why implementations of these algorithms are necessary. Index calculus is a complex algorithm that is even more complex when working with mathematical objects that are not easily represented by integers. Hence, the validity of a theoretical algorithm may be questioned as there are many subtleties that could be overlooked. Theoretical expositions often give complexity analyses to estimate the efficiency of their algorithm. These may be rigourous or based on heuristics regarding some unproven assumption. However, as computer scientists know, a complexity

analysis is often only an upper bound and hides numerical constants that may dramatically affect the algorithm's practical runtime. Moreover, an implementation is the only sure way to verify heuristic assumptions. Finally, implementations allow one to obtain concrete running times based on a variety of parameter choices. From these results we can extrapolate estimates on the time to complete larger problems. This is especially imporantant in the area of cryptography, where one chooses the size of their key based on the estimated time to complete the fastest known attack on the trapdoor function. This motivated us to study index calculus in the infrastructure of real quadratic function fields and to attempt the first implementation of this algorithm.

1.3 Contributions

In this thesis, we present the theory behind real quadratic function fields in such a way to show the parallels between algebraic geometry and pure algebraic constructions. We present algorithms for the infrastructure of real quadratic function fields and describe in detail how index calculus can be applied in this setting when the genus is large. We extend the algorithm of Müller, Stein and Thiel by describing numerous improvements to relation generation such as a baby walk method, self-initialized sieving, and the use of large primes. We give a new heuristic complexity analysis based on the baby walk method. Additionally, we present practical methods for computing the regulator that improve the linear algebra phase. We also descibe how to compute the class number, the group structure, and solve instances of the DLP in the infrastructure. Finally, we describe an implementation of our algorithms, the first time index calculus has been implemented in the infrastructure of real quadratic

function fields. We compared timings for two relation generation methods against an implementation of the standard baby-step giant-step method. Our results show that, as predicted, in high genus function fields index calculus is in fact significantly faster than the baby-step giant-step method for computing the regulator.

1.4 Organization

One goal of this thesis is to contain a complete introduction to the theory of behind the infrastructure of a real quadratic function field. The first chapter contains the necessary background on hyperelliptic curves and algebraic function fields. These two sections are meant to parallel each other, providing a geometric interpretation as well as a purely algebraic introduction. This background chapter may be skipped as the important parts are summarized at the beginning of Chapter 2.

Chapter 2 introduces the class groups of quadratic function fields and presents the theory behind the infrastructure of a real quadratic function field. We present algorithms and notation that is necessary for Chapter 3. Index calculus is introduced in Chapter 3. The chapter begins with a discussion of computational problems in the infrastructure of real quadratic function fields, followed by an overview of index calculus in a generic group. The main body of Chapter 3 is the discussion and complexity analysis of index calculus in the infrastructure.

Our C++ implementation of index calculus in the infrastructure of a real quadratic function field is described in Chapter 4. We discuss special improvements made in the implementation, such as sieving and using large primes, followed by details of the code and external libraries used. In this chapter we also present computation timings from our experiments. The last chapter contains conclusions and suggestions

for future work.

We have tried to present the algorithms in this thesis in a clear and precise manner. However, the reader should be aware that these high-level algorithms do not always correspond exactly to our C++ implementation due to optimizations and integration into a larger algebraic number theory library.

In this thesis we have attempted to give precise citations and complete references to assist readers who are looking for more information. Papers are cited by section, books by page number. The theorems we present are without proof, but we give the reference where a proof may be found. Otherwise, we have tried to give intuition and constructive arguments to support the theory presented.

We hope that you as a reader will find this thesis useful and easy to understand.

Chapter 2

Background

One can approach quadratic function fields from two different roads: the theory of algebraic geometry and the structures of non-geometric commutative algebra. However, one discovers that it is not that these roads meet at function fields, but they are in fact the same road, just different vehicles.

This chapter introduces the background theory from both of these vehicles. An attempt has been made to develop similar results in both areas, with a sensitivity to the theory and notation of each. Our motivation is that this chapter may serve as a resource to those familiar in one area to effectively communicate with a researcher in the other area regarding the details of what comes later in this thesis. It is our belief that this is the only such resource. Alternatively, since the results in each section are similar, a reader may choose to just review the one section that he/she is more comfortable with. Those not interested in this background theory may skip this chapter entirely since the important definitions are summarized at the beginning of Chapter 3.

To avoid starting from scratch, we assume that the reader is familiar with standard

commutative algebra, including rings, fields, ideals and valuations; definitions for this material can be found in any introductory text on the subject, *e.g.* Zariski and Samuel [ZS75].

An effort has been made to keep the number of references for the background material to a minimum. However, occasionally we do cite an alternative text if one of the following situations arises:

i) The general text does not contain the specific topic;

ii) The alternative text contains an exposition that is, in our opinion, more clear; or

iii) The treatment in alternative text is more standard or generalizes easier to other areas.

It is our intention that this will assist the reader who is interested in obtaining a more complete understanding of the concepts. Consequently, we ask forgiveness for not citing the original sources for this material.

2.1 Algebraic Geometric Background

Algebraic geometry has a beautiful theory of which, in this section, we only scratch the surface. In fact, we will only use concepts from classical algebraic geometry ignoring the powerful, yet cumbersome theory of schemes. We assume that the reader is familiar with affine and projective space. These concepts, as well as most of the material in this section, can be found in Fulton [Ful89] or Hartshorne [Har77].

2.1.1 Projective Plane Curves

Let K be a field and let $\overline{K}$ denote its algebraic closure. Let f be a non-constant polynomial in $K[x,y]$. The **affine hypersurface** defined by f is a set consisting of all the points (a,b) on the affine plane $\mathbb{A}^2(\overline{K})$ such that $f(a,b)=0$ [Ful89, pp. 7–8]. An **affine plane curve** C over K is a hypersurface defined by an irreducible polynomial $f \in K[x,y]$. The polynomial f generates a prime ideal $\mathfrak{p}_C \subset \overline{K}[x,y]$ that we associate with the curve C. The **coordinate ring** of C is the quotient ring

$$K[C] = K[x,y]/(\mathfrak{p}_C \cap K[x,y]) ,$$

and the field of quotients of $K[C]$, denoted $K(C) = \mathrm{Quot}(K[C])$, is called the **function field** of C over K. Similarly, one can define the function field of C over $\overline{K}$ as $\overline{K}(C) = \mathrm{Quot}(\overline{K}[x,y]/\mathfrak{p}_C)$. Elements of $K(C)$ (or $\overline{K}(C)$) are called **rational functions**. A rational function $f \in \overline{K}(C)$ is **regular** at a point $P \in C$ if there exists a pair of polynomials $g,h \in \overline{K}[C]$ such that $f = g/h$ and $h(P) \neq 0$. For any (finite) point $P \in C$, the **local ring** of C at P is the set $\mathcal{O}_P(C) \subseteq \overline{K}(C)$ of rational functions which are regular at the point P [Ful89, pp. 36, 42–43].

The set of K-rational points on the curve C is given by

$$\begin{aligned} C(K) &= \Big(C \cap \mathbb{A}^2(K)\Big) \cup \Big(C \cap H_\infty\Big) \\ &= \Big\{(a,b) \in C \;\Big|\; a,b \in K\Big\} \cup \Big(C \cap H_\infty\Big) , \end{aligned} \tag{2.1}$$

where H_∞ is the **hyperplane at infinity**. The curve C contains a non-empty subset of H_∞, elements which are referred to as **points at infinity**. These points at infinity do not exist on the affine plane, but their addition allows us to consider C in the

projective plane $\mathbb{P}^2(\overline{K})$. We convert points from affine space to projective space via the **homogenization** map:

$$\phi : \begin{cases} \mathbb{A}^2(\overline{K}) & \to & \mathbb{P}^2(\overline{K}) \\ (x, y) & \mapsto & (x : y : 1) \, . \end{cases}$$

Points of the form $(x : y : z) \in \mathbb{P}^2(\overline{K})$ are said to be in **homogeneous coordinates**. The **dehomogenization** map returns a homogeneous point $P = (x : y : z) \in \mathbb{P}^2(\overline{K})$ for $z \neq 0$ to $\phi^{-1}(P) = (x/z, y/z) \in \mathbb{A}^2(\overline{K})$; if $z = 0$, the dehomogenization map gives a point at infinity $\phi^{-1}(P) \in H_\infty$ [Ful89, pp. 86–87]. The affine curve C is said to be the **affine model** of the **projective plane curve** given by the homogenization map on C. For any projective plane curve we can find its affine model via the dehomogenization map. The function field $\overline{K}(C)$ of a projective plane curve C is isomorphic to the function field of its affine model [FL06, p. 51].

Let C be a projective plane curve over K. Let $\mathcal{O}_P(C)$ be the local ring of a point $P \in C$. The point P is called **smooth** if $\mathcal{O}_P(C)$ is integrally closed in $\overline{K}(C)$; otherwise, P is called **singular** [Har77, pp. 32, 40]. If every finite point on C is smooth, then the curve is said to be **nonsingular** (or **smooth**); otherwise, if there exists a singular finite point on C, the curve is **singular**. The **Jacobi criterion** assists in determining smooth points on C [Ful89, p. 64]:

Theorem 2.1 (Jacobi). *Let C be the affine model of a projective plane curve over a field K. Suppose C has a defining ideal $\mathfrak{p}_C$ generated by the polynomial $f \in K[x, y]$. For any point $P \in C$, if either of the partial derivatives $\frac{\partial f}{\partial x}(P) \neq 0$ or $\frac{\partial f}{\partial y}(P) \neq 0$, then P is smooth in both the projective curve and the affine model.*

It follows from Theorem 2.1 that a point P on a curve is singular if both of the partial

derivatives $\frac{\partial f}{\partial x}(P)$ and $\frac{\partial f}{\partial y}(P)$ vanish.

An example of a projective plane curve is the projective line $\mathbb{P}^1(K)$, called the **rational curve**. The rational curve has a unique point at infinity $P_\infty = \mathbb{P}^1(K) \cap H_\infty$. This point is smooth and has a local ring defined as

$$\mathcal{O}_\infty = \left\{ g/h \in K(\mathbb{P}^1) \;\middle|\; \deg g \leq \deg h \right\},$$

where $K(\mathbb{P}^1) = \mathrm{Quot}(K[x])$ is the function field of the rational curve [Ful89, p. 47]. In general, plane curves may have multiple points at infinity (which are not always smooth), but we can relate them to P_∞ by considering a "morphism" to the rational curve as defined in the next section.

2.1.2 Morphisms and Ramification Points

Let C and C' be two nonsingular projective curves over a field K. A non-constant map $\varphi : C' \to C$ is a morphism if there is a corresponding homomorphism $\widetilde{\varphi} : K(C) \to K(C')$. A morphism is either constant or surjective (we will ignore constant morphisms). A morphism $\varphi : C' \to C$ is **separable** if the function field $K(C')$ is a separable field extension of $K(C)$. The **degree** of φ is defined as $\deg \varphi = [K(C') : K(C)]$ [Ful89, p. 214].

Consider a point $P \in C'(\overline{K})$ with local ring $\mathcal{O}_P$. The local ring is a discrete valuation ring. Let $t \in \mathcal{O}_P$ be a uniformizing parameter, *i.e.* t generates the maximal ideal $\mathfrak{m}_P \subset \mathcal{O}_P$. Then every $f \in \overline{K}(C')$ can be expressed uniquely as $f = t^n u$ for some $u \in \mathcal{O}_P^*$ and $n \in \mathbb{Z}$. The value of n is independent of the choice of t. Then we define the (discrete[1]) **valuation** of $\mathcal{O}_P$ as $v_P(f) = n$ and $v_P(0) = \infty$. This valuation

[1] "Discrete" implies that the result of the valuation is an integer [ZS75, V. II, p. 42].

can be extended to the function field $\overline{K}(C')$ by the rule $v_P(g/h) = v_P(g) - v_P(h)$ [Ful89, pp. 46–47].

Let $P \in C'(\overline{K})$ and let t be a uniformizing parameter of $\mathcal{O}_Q$, where $Q = \varphi(P)$. The **ramification index** of φ at P is the positive integer given by

$$e(P) = v_P(t) ,$$

where v_P is the valuation of the local ring $\mathcal{O}_P$. The ramification index is independent of the choice of t and satisfies $e(P) = 1$ for all but a finite number of points $P \in C'$. The points $P \in C'$ such that $\varphi(P) \in C$ and $e(P) > 1$ are called **ramification points** [Ful89, pp. 214–215].

Consider a ramification point $P \in C'$ with index $e(P)$. If $\operatorname{char} K = 0$ or if $\operatorname{char} K = p$ and $p \nmid e(P)$, then P is said to be **tamely ramified** with respect to φ. Otherwise, if $\operatorname{char} K = p$ and $p | e(P)$, then P is **wildly ramified** [Har77, p. 299].

2.1.3 Divisors

Let C be a nonsingular projective (plane) curve over K. A **divisor** on C is a formal sum of points,

$$D = \sum_{P \in C(\overline{K})} n_P P , \tag{2.2}$$

where $n_P \in \mathbb{Z}$ with finitely many n_P non-zero. Points on C are **prime divisors** and we denote the coefficients of the prime divisors of D as $\operatorname{ord}_P(D) = n_P$. The **support** of D, denoted $\operatorname{supp}(D)$, is the set of points $P \in C$ with non-zero coefficients $\operatorname{ord}_P(D)$. If $\operatorname{ord}_P(D) \geq 0$ for all $P \in \operatorname{supp}(D)$, then D is **effective**, denoted $D \geq 0$.

The **degree** of D is defined as

$$\deg D = \sum_{P \in \mathrm{supp}(D)} \mathrm{ord}_P(D) \, .$$

The set of divisors on C forms a free Abelian group $\mathrm{Div}(C)$ under addition [Har77, p. 294].

Recall that for any point $P \in C(\overline{K})$, the local ring $\mathcal{O}_P$ is a discrete valuation ring with valuation v_P. Let $f \in \overline{K}(C)$ be a non-zero rational function. The **divisor** of f is defined as

$$\mathrm{div}(f) = \sum_{P \in C(\overline{K})} v_P(f) \cdot P \, , \tag{2.3}$$

where $v_P(f)$ is non-zero for a finite number of points $P \in C(\overline{K})$. Since the valuation v_P is discrete, it is easy to see that (2.3) satisfies the form given in (2.2). Divisors that can be written in this form are called **principal divisors**. Furthermore, we have that $\mathrm{div}(f \cdot g) = \mathrm{div}(f) + \mathrm{div}(g)$ and $\mathrm{div}(f/g) = \mathrm{div}(f) - \mathrm{div}(g)$ for any non-zero $f, g \in \overline{K}(C)$ [Har77, pp. 130–131].

For any principal divisor $\mathrm{div}(f)$, we define the **divisor of zeroes** $\mathrm{div}_0(f)$ and the **divisor of poles** $\mathrm{div}_\infty(f)$ given by

$$\mathrm{div}_0(f) = \sum_{\substack{P \in C(\overline{K}) \\ v_P(f) > 0}} v_P(f) \cdot P \qquad \text{and} \qquad \mathrm{div}_\infty(f) = \sum_{\substack{P \in C(\overline{K}) \\ v_P(f) < 0}} -v_P(f) \cdot P \, .$$

Then $\mathrm{div}(f)$ can be written as $\mathrm{div}(f) = \mathrm{div}_0(f) - \mathrm{div}_\infty(f)$. Any point $P \in \mathrm{supp}(\mathrm{div}_0(f))$ is a **zero** of f of multiplicity $v_P(f)$. Similarly, any point $P \in \mathrm{supp}(\mathrm{div}_\infty(f))$ is a **pole**

of f of order $-v_P(f)$. Every principal divisor has degree zero; therefore, every rational function f has an equal number of zeroes and poles when counted with multiplicities [Ful89, p. 188].

Let C and C' be two nonsingular projective curves over K with a morphism $\varphi : C' \to C$. The **conorm** of a divisor with respect to φ is the following homomorphism:

$$\mathrm{Con}_\varphi : \begin{cases} \mathrm{Div}(C) & \to & \mathrm{Div}(C') \\ D & \mapsto & \displaystyle\sum_{Q \in \mathrm{supp}(D)} \sum_{\substack{P \in C'(\overline{K}) \\ \varphi(P)=Q}} \mathrm{ord}_Q(D) \cdot e(P) \cdot P \,, \end{cases}$$

where $e(P)$ is the ramification index of φ at P. In particular, for any divisor $D \in \mathrm{Div}(C)$, the degree of the divisor $\mathrm{Con}_\varphi(D) \in \mathrm{Div}(C')$ satisfies

$$\deg \mathrm{Con}_\varphi(D) = \deg \varphi \cdot \deg D \,, \tag{2.4}$$

where $\deg \varphi$ is the degree of the morphism φ [Har77, pp. 137–138].

2.1.4 Differentials and Canonical Divisors

Let $F = K(C)$ be the function field of a nonsingular projective (plane) curve over K. Note that F is a field containing the field K and let V be a vector space over F. A **derivation** of F into V is a map $\mathrm{d} : F \to V$ satisfying the following properties for every $f, g \in F$ and $a \in K$:

i) $\mathrm{d}(fg) = f \cdot \mathrm{d}(g) + g \cdot \mathrm{d}(f)$ (Leibniz's rule)

ii) $\mathrm{d}(f+g) = \mathrm{d}(f) + \mathrm{d}(g)$

iii) $\mathrm{d}(af) = a\mathrm{d}(f)$.

For each $f \in F$ let $[f]$ denote a formal symbol. We define the free vector space $S = \left\{ [f] \;\middle|\; f \in F \right\}$ over F and a subspace T generated by the following sets:

i) $\left\{ [fg] - f[g] - g[f] \;\middle|\; f, g \in F \right\}$

ii) $\left\{ [f+g] - [f] - [g] \;\middle|\; f, g \in F \right\}$

iii) $\left\{ [af] - a[f] \;\middle|\; f \in F, a \in K \right\}$.

Then the **space of differentials** of $F = K(C)$ is defined to be the quotient space $\Omega_{F/K} = S/T$. The residue of $[f]$ in S/T is the image of the derivation $\mathrm{d}(f)$ in $\Omega_{F/K}$. We will omit the parenthesis and write $\mathrm{d}f$ for the derivation $\mathrm{d}(f)$. Elements of $\Omega_{F/K}$ are called **differentials** on C [Ful89, pp. 203–204].

For any (smooth) point $P \in C(K)$, consider the local ring $\mathcal{O}_P(C)$ as a discrete valuation ring and subring of $F = K(C)$. Let $t \in \mathcal{O}_P(C)$, $t \notin K$, be a uniformizing parameter. Then, for every $\omega \in \Omega_{F/K}$, there is a unique element $h \in F$, depending on ω and t, such that $\omega = h \cdot \mathrm{d}t$. Since $\omega = \mathrm{d}f$ for some $f \in F$, the value h can be written as

$$h = \frac{\mathrm{d}f}{\mathrm{d}t} ,$$

where h is called the **derivative** of f with respect to t. The **order** of ω at P is given by the valuation on the local ring $\mathcal{O}_P$ as

$$\mathrm{ord}_P(\omega) = v_P(\omega/\mathrm{d}t) .$$

As the notation suggests, the order depends only on ω and P, not on the choice of uniformizer t [Ful89, p. 207].

For any differential $\omega \in \Omega_{F/K}$, the **divisor** of ω is a formal sum given by

$$\operatorname{div}(\omega) = \sum_{P \in C} \operatorname{ord}_P(\omega) \cdot P \, .$$

Note that $\operatorname{div}(\omega) \in \operatorname{Div}(C)$ and divisors of this form are called **canonical divisors** [Ful89, p. 207].

2.1.5 Riemann-Roch and the Genus

For each divisor $D \in \operatorname{Div}(C)$, the **Riemann-Roch space** is a vector space $\mathscr{L}(D)$ over K given by

$$\mathscr{L}(D) = \left\{ f \in K(C) \setminus \{0\} \;\middle|\; \operatorname{div}(f) + D \geq 0 \right\} \cup \{0\} \, .$$

The Riemann-Roch space is finite-dimensional, and its dimension is denoted by

$$\ell(D) = \dim_K \mathscr{L}(D)$$

[Ful89, p. 192]. The Riemann-Roch space is important as shown in the famous **Riemann-Roch theorem** [Ful89, pp. 209–210]:

Theorem 2.2 (Riemann, 1857; Roch, 1865). *There exists an integer $g \geq 0$ satisfying*

i) $\ell(D) - \ell(W - D) = \deg(D) - g + 1$ *for any divisor* $D \in \operatorname{Div}(C)$ *and canonical divisor* $W \in \operatorname{Div}(C)$.

ii) $\ell(D) = \deg(D) - g + 1$ *for any divisor* $D \in \operatorname{Div}(C)$ *with* $\deg D > 2g - 2$.

The value g is called the **genus** of the curve C. We call any nonsingular projective curve of genus $g = 0$ a **rational curve** since it is (birationally[2]) equivalent to $\mathbb{P}^1(\overline{K})$. A nonsingular projective curve of genus $g = 1$ is called an **elliptic curve** [Ful89, pp. 196–198]. We consider curves of higher genus in Section 2.1.7, but first we give another method for computing the genus.

2.1.6 The Ramification Divisor and the Hurwitz Formula

Let C and C' be two nonsingular projective curves over a field K with a separable morphism $\varphi : C' \to C$. To simplify notation, denote the function fields of C and C' as $F = K(C)$ and $F' = K(C')$, respectively. Since there is an injective homomorphism $\tilde{\varphi} : F \to F'$ corresponding to φ, $\tilde{\phi}(F)$ is isomorphic to F and, hence, F is a subfield of F'.

Consider the modules $\Omega_{F/K}$ and $\Omega_{F'/K}$. There is a **module of relative differentials** $\Omega_{F'/F}$ of F' over F defined in a similar way as $\Omega_{F'/K}$ with $K \subset F'$ replaced by $F \subset F'$. Let $\mathrm{Div}(\Omega_{F/K}) \subset \mathrm{Div}(C)$ denote the set of canonical divisors of C. Then we have the following exact sequence [Har77, pp. 172, 299–300]:

$$0 \longrightarrow \mathrm{Con}_{\varphi}(\mathrm{Div}(\Omega_{F/K})) \longrightarrow \Omega_{F'/K} \longrightarrow \Omega_{F'/F} \longrightarrow 0\,. \tag{2.5}$$

Consider a point $P \in C'$ with $\varphi(P) = Q$ for some point $Q \in C$. Let $(\Omega_{F'/K})_P$ denote the free $\mathcal{O}_P$-module generated by a differential $\mathrm{d}t$, where $t \in \mathcal{O}_P$ is a uniformizing parameter. Similarly, let $(\Omega_{F/K})_Q$ denote the free $\mathcal{O}_Q$-module generated by $\mathrm{d}u$ for a uniformizing parameter $u \in \mathcal{O}_Q$. Then from the sequence in Equation (2.5),

[2]Two curves are "birationally" equivalent if and only if their function fields are isomorphic [Ful89, p. 155].

there is a principal $\mathcal{O}_P$-module $(\Omega_{F'/F})_P$ generated by $\mathrm{ord}_P(\mathrm{d}u/\mathrm{d}t)$ that is isomorphic to $(\Omega_{F'/K})_P/\operatorname{Con}_\varphi(\mathrm{Div}(\Omega_{F/K})_Q)$. Let $\mathrm{length}(M)$ denote the length of a module M; that is, the length of the longest ascending chain of submodules of M. Then we have the following result regarding the length of $(\Omega_{F'/F})_P$ [Har77, p. 300]:

Lemma 2.3. *Let C and C' be two nonsingular projective curves over a field K with a separable morphism $\varphi : C' \to C$. Then for any point $P \in C'(\overline{K})$*

$$\mathrm{length}\,(\Omega_{F'/F})_P \geq e(P) - 1\,.$$

If P is tamely ramified with respect to φ, then the inequality is an equality; otherwise, P is wildly ramified and the inequality is strict.

Given the previous result, we define the **ramification divisor** of φ as

$$R_\varphi = \sum_{P \in C'(\overline{K})} \mathrm{length}\,(\Omega_{F/F'})_P \cdot P\,. \tag{2.6}$$

The points with non-zero coefficients in R_φ are precisely the ramification points of φ. This leads to the **Hurwitz formula** which relates the genera of two curves [Har77, p. 301].

Theorem 2.4 (Hurwitz, 1891). *Let C and C' be two nonsingular projective curves over a field K with a separable morphism $\varphi : C' \to C$. Then if C has genus g, the genus g' of C' satisfies*

$$2g' - 2 = \deg(\varphi)(2g - 2) + \deg R_\varphi\,.$$

From the definition of the ramification divisor in Equation (2.6) and Lemma 2.3,

we get the following corollary to the Hurwitz formula [Sil86, p. 41]:

Corollary 2.5. *Let C and C' be two nonsingular projective curves over a field K with a separable morphism $\varphi : C' \to C$. Then if C has genus g, the genus g' of C' satisfies*

$$2g' - 2 \geq \deg(\varphi)(2g-2) + \sum_{P \in C'(\overline{K})} \big(e(P) - 1\big) \, .$$

If all $P \in C'$ are tamely ramified, then the inequality is an equality; otherwise, the inequality is strict.

2.1.7 Weierstrass Points

Let C be a nonsingular projective curve of genus g defined over a field K and let $\mathbb{P}^1$ be the rational curve also over K. A morphism $\pi : C \to \mathbb{P}^1$ is called a **cover** for C and the ramification points with respect to π are called **Weierstrass points** of C. The **fiber** of π over P is the set $\pi^{-1}(P)$ of points $Q \in C$ such that $\pi(Q) = P$. For a cover π of degree n, if the fiber contains fewer than n points, then it must contain at least one Weierstrass point of C.

The **Weierstrass gap sequence** for a point $P \in C$ is given by the integers $n \in \mathbb{N}$ satisfying

$$\ell(nP) = \ell\big((n-1)P\big) \, .$$

There are exactly g integers in the Weierstrass gap sequence $(n_1, n_2, \ldots, n_g)$, satisfying $0 < n_1 < n_2 < \cdots < n_g < 2g$. All but finitely many points $P \in C(\overline{K})$ have the same

Weierstrass gap sequence; those points that differ are precisely the Weierstrass points. There are no Weierstrass points on curves of genus $g = 0$ or $g = 1$; however, if $g \geq 2$, then there must exist at least one Weierstrass point [Sti93, p. 32].

If $g > 1$ and 2 is not in the Weierstrass gap sequence for a Weierstrass point P, then P is called a **hyperelliptic Weierstrass point**. A nonsingular projective curve C is called a **hyperelliptic curve** if C has a hyperelliptic Weierstrass point. This is satisfied if and only if the genus $g \geq 2$ and there exists a **double cover** for C, *i.e.* a morphism $\pi : C \to \mathbb{P}^1$ of degree 2 [Ful89, p. 216].

2.1.8 Hyperelliptic Curves

Recall from the previous section that a hyperelliptic curve C over a field K is a nonsingular projective curve of genus $g > 1$ with a morphism $\pi : C \to \mathbb{P}^1$ of degree 2. Let $K[C]$ be the coordinate ring of C given by $K[C] = K[x, y]/\mathfrak{p}_C$, where $\mathfrak{p}_C$ is the prime ideal of the curve C. Let $P_\infty \in C \cap H_\infty$ be a point at infinity on $C(K)$ (*cf.* Equation (2.1)). In this section we derive the equation defining the nonsingular affine model of a hyperelliptic curve C/K following the technique of Frey and Lange [FL06, pp. 73–74].

There exists an element x that is transcendental over K such that the rational curve $\mathbb{P}^1/K$ has a function field $K(\mathbb{P}^1) = K(x)$. For such an x, consider the divisor of poles $D = \mathrm{div}_\infty(x)$. Since π has degree 2, we have $[K(C) : K(x)] = 2$ and the divisor D must have degree 2. For integers $1 \leq m \leq g$ we have $\deg(mD) = 2m$ and $\mathscr{L}(mD)$ has a basis $\{1, x, \ldots, x^m\}$. But when $m = g + 1$ we have $\deg\big((g+1)D\big) = 2(g+1)$,

and by Riemann-Roch (Theorem 2.2)

$$\ell\big((g+1)D\big) = \deg\big((g+1)D\big) - g + 1 = g + 3\,.$$

This implies that $\mathscr{L}\big((g+1)D\big)$ has a basis consisting of $1, x, \ldots, x^{g+1}$ plus some additional linearly independent $z \in \mathscr{L}\big((g+1)D\big)$. We will show that $z \notin K[x]$ by first supposing that $z = x^{g+2}$. Then there would be $g+2$ zeroes of z implying $\deg \mathrm{div}_0(z) \geq g+2$. In fact, given that π is a morphism of degree 2, by Equation (2.4) we would have that $\deg \mathrm{div}_0(z) \geq 2(g+2)$. But $\deg \mathrm{div}_\infty(z) = \deg \mathrm{div}_0(z)$ and $\mathrm{div}(z) = \mathrm{div}_0(z) - \mathrm{div}_\infty(z)$, so with $\deg\big((g+1)D\big) = 2g+2$ we could not have $\mathrm{div}(z) + (g+1)D$ be effective. Therefore, $x^{g+2} \notin \mathscr{L}\big((g+1)D\big)$ and $z \notin K[x]$ implying the basis for $\mathscr{L}\big((g+1)D\big)$ is $\{1, x, \ldots, x^{g+1}, y\}$. In a similar vein, the Riemann-Roch space $\mathscr{L}\big(2(g+1)D\big)$ has a basis of size $3g+5$ given by the following:

$$\{1, x, \ldots, x^{2(g+1)}, y, xy, \ldots, x^{g+1}y, y^2\}\,.$$

A linear combination of this basis over K gives the following equation (after dividing through by the coefficient of y^2):

$$y^2 + h(x)y = f(x)\,, \tag{2.7}$$

where $h, f \in K[x]$ with degrees bounded as $\deg h \leq g+1$ and $\deg f \leq 2g+2$.

Fields of Odd Characteristic

Now we present more precise requirements on the degrees of h and f that depend on the number of finite Weierstrass points on C. We will start by assuming that $\operatorname{char} K \neq 2$. Since π is a separable morphism of degree 2, Corollary 2.5 of the Hurwitz formula simplifies to

$$2g + 2 = \sum_{P \in C(\overline{K})} \big(e(P) - 1\big) . \tag{2.8}$$

where each ramification index $e(P) \leq 2$. Therefore, there are at most $2g + 2$ ramification points on C with respect to φ. However, if $P_\infty \in C$ is a ramification point, then there are only $2g + 1$ finite ramification points. These ramification points are precisely the Weierstrass points of C.

The transformation $y \mapsto y - h(x)/2$ allows us to consider $h(x) = 0$ in Equation (2.7). Then there exists an involution given by $\iota : y \mapsto -y$, called the **hyperelliptic involution**. This involution ι sends a point P to the other point in the fiber of $\pi(P)$ and fixes the ramification points, *i.e.* the Weierstrass points of C. The image $\iota(P)$ of a point P under the hyperelliptic involution is called the **conjugate** of P, denoted $\overline{P}$.

If there is one point in the fiber of $\pi(P_\infty)$, then P_∞ is the only point at infinity in $C \cap H_\infty$. If P_∞ is a Weierstrass point, then by Equation (2.8) there are $2g + 1$ finite Weierstrass points and we call C an **imaginary hyperelliptic curve**. In this case $f(x)$ is monic with $\deg f = 2g + 1$. An example of an imaginary hyperelliptic curve is shown in Figure 2.6.

If there are two points in the fiber of $\pi(P_\infty)$, then there are two points at infinity,

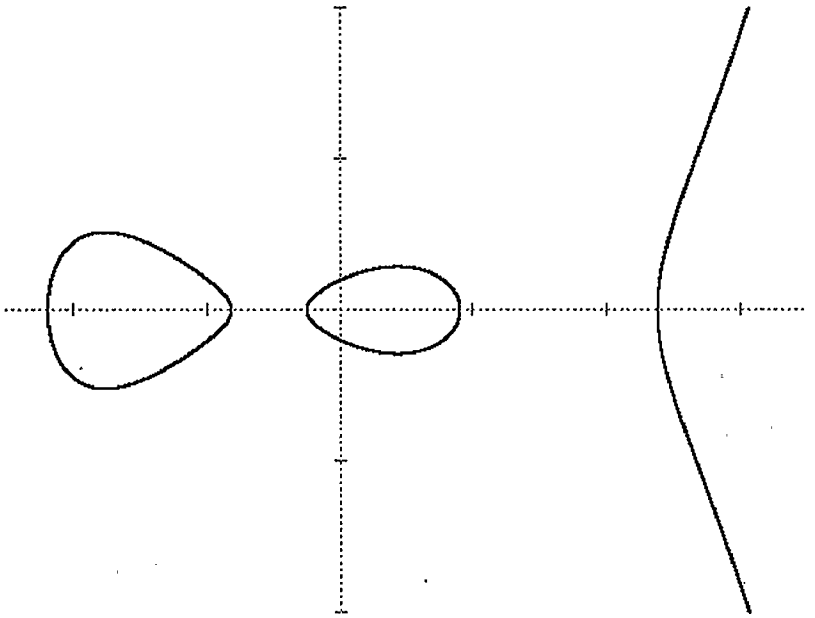

Figure 2.6. *The genus-2 imaginary hyperelliptic curve* $C : y^2 = x^5 - 6x^3 - x^2 + 4x + 1$ *over* $\mathbb{R}$ *plotted in the affine plane* $\mathbb{A}^2(\mathbb{R})$.

$P_\infty, P'_\infty \in C \cap H_\infty$, such that $\overline{P_\infty} = P'_\infty$. Also, since these points are not Weierstrass points, by Equation (2.8) there are $2g + 2$ finite Weierstrass points and C is called a **real hyperelliptic curve**. In this case $f(x)$ has $\deg f = 2g+2$ with leading coefficient a square in K. An example of a real hyperelliptic curve is shown in Figure 2.7.

There is a degenerate case when there is only one point in the fiber of $\pi(P_\infty)$, but the point P_∞ is not a Weierstrass point. In the degenerate case, C is called **unusual** and $f(x)$ has $\deg f = 2g + 2$ with leading coefficient a non-square in K.

These cases give the affine equation $C : y^2 = f(x)$ for a hyperelliptic curve C over K ($\operatorname{char} K \neq 2$), where $\deg f \in \{2g+1, 2g+2\}$. The requirements on the polynomial f for each of the above cases can be found in Jacobson, Scheidler and Stein [JSS07b, §3] or [JSS07a, §2]. To satisfy the Jacobi criterion (Theorem 2.1) and ensure C is

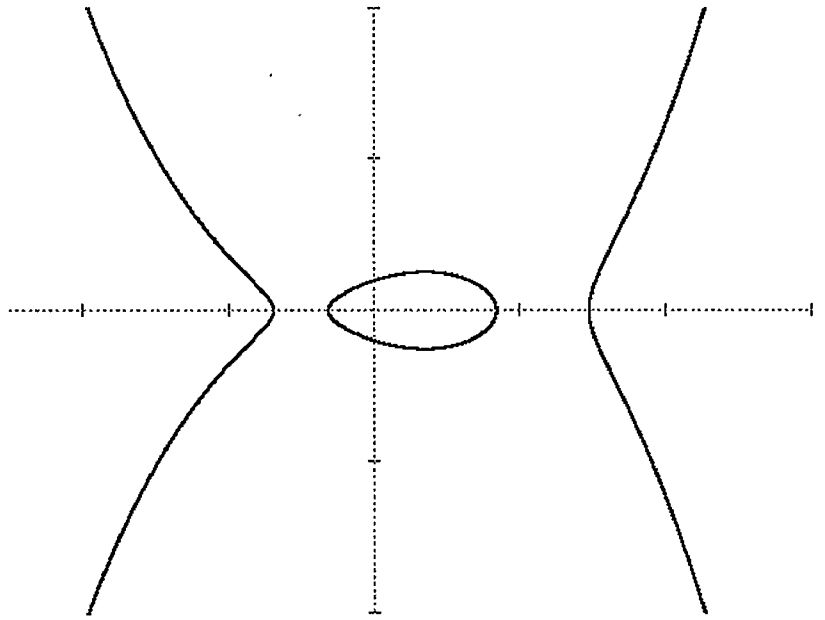

Figure 2.7. *The genus-2 real hyperelliptic curve* $C : y^2 = x^6 + x^4 - 5x^3 - 2x^2 + 3x + 1$ *over* $\mathbb{R}$ *plotted in the affine plane* $\mathbb{A}^2(\mathbb{R})$.

nonsingular, no finite point $P \in C(\overline{K})$ must satisfy $2y = 0$ and $f'(x) = 0$. That is, the singular points are $(a, 0) \in C(\overline{K})$ satisfying $f(a) = 0$ and $f'(a) = 0$. Therefore, C is nonsingular if and only if $f(x)$ is squarefree in $\overline{K}[x, y]$.

Fields of Even Characteristic

Now we consider the case when K is of characteristic two. With the degree 2 separable morphism π, the Hurwitz formula (Theorem 2.4) simplifies to

$$2g + 2 = \deg R_\varphi \, .$$

Let $F_C = K(C)$ and $F = K(\mathbb{P}^1) = K(x)$. Since every ramified point $P \in C(\overline{K})$ is wildly ramified, Lemma 2.3 gives that length $(\Omega_{F_C/F})_P \geq 2$. Therefore, from the definition of the ramification divisor in Equation (2.6), the number r of ramified points on C with respect to φ must be in the range $1 \leq r \leq g+1$. These ramification points are precisely the Weierstrass points of C.

We cannot use the same transformation $y \mapsto y - h(x)/2$ when $\operatorname{char} K = 2$. If we were to let $h(x) = 0$, then C would be singular since any Weierstrass point $(a, b) \in C(\overline{K})$ satisfies $f'(a) = 0$ and $2b = 0$. Therefore, we must have $h(x) \neq 0$ in (2.7). The **hyperelliptic involution** for this case is given by $\iota : y \mapsto -y - h(x)$. Again, $\overline{P} = \iota(P)$ is called the **conjugate** of P, and the fixed points of ι are the Weierstrass points of C.

If there is one point in the fiber of $\pi(P_\infty)$, then P_∞ is the only point at infinity in $C \cap H_\infty$. If P_∞ is a Weierstrass point and there are at most g finite Weierstrass points on C, then we call C an **imaginary hyperelliptic curve**. In this case $\deg h \leq g$ and $\deg f = 2g + 1$.

If there are two points in the fiber of $\pi(P_\infty)$, then there are two points at infinity, $P_\infty, P'_\infty \in C \cap H_\infty$, such that $\overline{P_\infty} = P'_\infty$. These points are not Weierstrass points, so if there are $g+1$ finite Weierstrass points, we call C a **real hyperelliptic curve**. In this case we have $\deg h = g + 1$ and $f(x)$ satisfies either $\deg f \leq 2g + 1$ or $\deg f = 2g + 2$ with leading coefficient of the form $e^2 + e$ for some $e \in K^*$.

We still have the degenerate case where C is called **unusual** when there is only one point in the fiber of $\pi(P_\infty)$, but the point P_∞ is not a Weierstrass point. In this case $\deg h = g + 1$ and $\deg f = 2g + 2$ where $f(x)$ does not have a leading coefficient of the form $e^2 + e$ for any $e \in K^*$.

Therefore, when $\operatorname{char} K = 2$, we have an affine equation $C : y^2 + h(x)y = f(x)$ for a hyperelliptic curve C over K, where $h(x)$ is monic and satisfies $\deg h \leq g + 1$ and $f(x)$ satisfies $\deg f \in \{2g + 1, 2g + 2\}$. It is interesting to note that the irreducible factors of $h(x)$ are simple divisors of $f(x)$. The requirements on the polynomials h and f for each of the above cases are derived in Enge [Eng01]. To satisfy the Jacobi criterion (Theorem 2.1) and ensure C is nonsingular, no finite point $P \in C(\overline{K})$ can satisfy both $2y + h(x) = 0$ and $f'(x) - h'(x)y = 0$.

In the rest of this thesis we ignore the case where C is unusual. One may note that if considering a curve C that is unusual with constant field K, then C over L, where L is a constant field extension of degree 2 over K, will result in C being a real hyperelliptic curve [PR99, §1]. The focus of this thesis is real hyperelliptic curves and their function fields.

2.2 Algebraic Non-Geometric Background

Commutative algebra offers mathematical structures that work in similar ways to the geometric presentation of the last section. In particular, this section defines and examines the construct of an algebraic function field. As a reference for most of the results in this section, we refer the reader to Stichtenoth [Sti93].

Let K be a field. An **algebraic function field** (in one variable) over K is a field F that is a finite algebraic extension of $K(x)$ for some $x \in F$ that is transcendental over K. The algebraic function field $F = K(x)$ is called the **rational function field** in x over K. Throughout this section we assume that K is a perfect field.[3]

[3] A field K is "perfect" if every irreducible polynomial $f(x) \in K[x]$ is separable over K [ZS75, V. I, pp. 64–65]. Finite fields, fields of characteristic zero, and algebraically closed fields are all perfect.

The **field of constants** $\widetilde{K}$ of an algebraic function field F over K is the set of all elements of F that are algebraic over K. We have that $K \subseteq \widetilde{K} \subset F$ and $\widetilde{K}$ is a finite extension field of K. The field of constants is **exact** (or **full**) if $\widetilde{K} = K$ [Sti93, pp. 1, 6]. We will assume that any function field F has $\widetilde{K} = \overline{K}$.

The theory of algebraic function fields has many parallels with algebraic number fields. In fact, number fields and function fields defined over finite fields are generally referred to as **global fields**. In the work at hand, we focus on algebraic function fields.

2.2.1 Places

Let $F \supset K$ be an algebraic function field. A **place** of F is an embedding $\mathscr{P}$ of a valuation ring $\mathcal{O}_{\mathscr{P}} \subset F$ into the residue field $\mathcal{O}_{\mathscr{P}}/\mathfrak{m}_{\mathscr{P}}$, where $\mathfrak{m}_{\mathscr{P}}$ is the maximal ideal of $\mathcal{O}_{\mathscr{P}}$. The value of $\mathscr{P}$ at an element $a \in \mathcal{O}_{\mathscr{P}}$ is written in infix notation and a place $\mathscr{P}$ satisfies the following properties:

i) For any $a \in F \setminus \mathcal{O}_{\mathscr{P}}$, we have $a^{-1} \in \mathcal{O}_{\mathscr{P}}$ and $a^{-1}\mathscr{P} = 0$.

ii) There exists an $a \in \mathcal{O}_{\mathscr{P}}$ such that $a\mathscr{P} \neq 0$.

The map $\mathscr{P}$ can be extended to F by introducing a symbol ∞ such that $a\mathscr{P} = \infty$ for all $a \in F \setminus \mathcal{O}_{\mathscr{P}}$. The **degree** of a place $\mathscr{P}$ is a finite value given by

$$\deg \mathscr{P} = [\mathcal{O}_{\mathscr{P}}/\mathfrak{m}_{\mathscr{P}} : K] \leq [F : K] < \infty .$$

If $\deg \mathscr{P} = 1$, then $\mathcal{O}_{\mathscr{P}}/\mathfrak{m}_{\mathscr{P}} = K$ and the place $\mathscr{P}$ is called **rational**. We denote the set of places of F by $\Sigma(F)$ [ZS75, V. II, pp. 3–7], [Sti93, pp. 6,7].

A place $\mathscr{P} \in \Sigma(F)$ is **finite** on $K[x]$ if $a\mathscr{P} \neq \infty$ for all $a \in K[x]$. The **centre** of a finite place $\mathscr{P}$ is the prime ideal $\mathfrak{p} = \mathfrak{m}_{\mathscr{P}} \cap K[x] \subseteq K[x]$, consisting of all the polynomials $f \in K[x]$ that vanish at P. Two places are **isomorphic** if they have the same centre in any integral domain $R \subset F$ in which they are finite [ZS75, V. II, pp. 15–16].

Consider a place $\mathscr{P} \in \Sigma(F)$ with valuation ring $\mathcal{O}_{\mathscr{P}}$ and maximal ideal $\mathfrak{m}_{\mathscr{P}}$. For any prime element t satisfying $\mathfrak{m}_{\mathscr{P}} = t\mathcal{O}_{\mathscr{P}}$, there exists a unique representation $f = t^n u$ for every non-zero $f \in F$ with $u \in \mathcal{O}_{\mathscr{P}}^*$ and $n \in \mathbb{Z}$. Note that the value of n is independent of the choice of t. This defines the (discrete) **valuation** of $\mathcal{O}_{\mathscr{P}}$ given by $v_{\mathscr{P}}(f) = n$ and $v_{\mathscr{P}}(0) = \infty$. Then the valuation ring $\mathcal{O}_{\mathscr{P}}$ for any finite place $\mathscr{P}$ is given as the following [Sti93, pp. 3–5]:

$$\mathcal{O}_{\mathscr{P}} = \left\{ f \in F \;\middle|\; v_{\mathscr{P}}(f) \geq 0 \right\}.$$

Let $K(x)$ be the rational function field in x over K. There is a unique place $\mathscr{P}_\infty \in \Sigma(K(x))$ not finite on $K[x]$ called the **place at infinity**. This place has a discrete valuation defined by the rule $v_{\mathscr{P}_\infty}(g/h) = \deg h - \deg g$, for any $f = g/h \in F$. Then the valuation ring of $\mathscr{P}_\infty$ is given by

$$\mathcal{O}_\infty = \left\{ g/h \in K(x) \;\middle|\; \deg g \leq \deg h \right\}.$$

The transcendental element $x \notin \mathcal{O}_\infty$, so we must have $x^{-1} \in \mathcal{O}_\infty$. Therefore, $K[x^{-1}] \subseteq \mathcal{O}_\infty$ and the centre of $\mathscr{P}_\infty$ is the principal (prime) ideal $\mathfrak{p} = x^{-1}\mathcal{O}_\infty$. The place at infinity satisfies $\deg \mathscr{P}_\infty = 1$. Any algebraic function field that is a non-trivial extension of $K(x)$ may actually have multiple places at infinity, but they

are all related to $\mathscr{P}_\infty$ [Sti93, pp. 8–9].

2.2.2 Extensions and Ramification of Places

Let F' denote an algebraic function field over K that is a finite algebraic extension of the rational function field $F = K(x)$. Let $\mathscr{P} \in \Sigma(F)$ and $\mathscr{P}' \in \Sigma(F')$ be places, and $\mathcal{O}_{\mathscr{P}}, \mathcal{O}_{\mathscr{P}'}$ their respective valuation rings. The place $\mathscr{P}'$ is an **extension** of $\mathscr{P}$, or equivalently $\mathscr{P}'$ **lies above** $\mathscr{P}$, if $\mathcal{O}_{\mathscr{P}} \subseteq \mathcal{O}_{\mathscr{P}'}$. If $\mathscr{P}'$ lies above $\mathscr{P}$, this is often denoted as $\mathscr{P}'|\mathscr{P}$. Each place $\mathscr{P}' \in \Sigma(F')$ lies above one unique place $\mathscr{P} \in \Sigma(F)$. Conversely, each place $\mathscr{P} \in \Sigma(F)$ has at least one extension in $\Sigma(F')$. If two places are extensions of each other, then the places are isomorphic [Sti93, pp. 60–62].

Two places $\mathscr{P}'_1, \mathscr{P}'_2 \in \Sigma(F')$ are said to be **conjugate** over F if there exists an F-automorphism σ of F' such that $\mathscr{P}'_2 = \sigma\mathscr{P}'_1$. If $\mathscr{P}'_1$ and $\mathscr{P}'_2$ both lie above $\mathscr{P}$, then $\mathscr{P}'_2$ is isomorphic to a conjugate of $\mathscr{P}'_1$. We write $\mathscr{P}'_2 = \overline{\mathscr{P}'_1}$ to denote that $\mathscr{P}'_2$ is a conjugate (or isomorphic to a conjugate) of $\mathscr{P}'_1$ [ZS75, V. II, p. 28].

Let $\mathscr{P}' \in \Sigma(F')$ be a place lying above $\mathscr{P} \in \Sigma(F)$. There exists an integer $e \in \mathbb{N}$ such that, for any $f \in F'$,

$$v_{\mathscr{P}'}(f) = e \cdot v_{\mathscr{P}}(f) \, .$$

This integer $e = e(\mathscr{P}'|\mathscr{P})$ is called the **ramification index** of $\mathscr{P}'$ over $\mathscr{P}$. The place $\mathscr{P}$ is said to be **ramified** in F' (or $\mathscr{P}'$ is **ramified** over $\mathscr{P}$) if $e > 1$; otherwise, $\mathscr{P}$ (resp. $\mathscr{P}'$) is **unramified** [Sti93, pp. 61–62].

Let $\mathscr{P}' \in \Sigma(F')$ be a ramified place lying above $\mathscr{P} \in \Sigma(F)$. Let $p = \mathrm{char}(K)$. If either $p = 0$ or $p \nmid e(\mathscr{P}'|\mathscr{P})$, then $\mathscr{P}$ is **tamely ramified** in F'; otherwise, $\mathscr{P}$ is

wildly ramified. The function field F' is said to be a **tame extension** of F if all the ramified places are tamely ramified [Sti93, pp. 94–95].

2.2.3 Divisors on Function Fields

Let F be an algebraic function field. A **divisor** on F is given by the formal sum

$$D = \sum_{\mathscr{P} \in \Sigma(F)} n_{\mathscr{P}} \mathscr{P},$$

where $n_{\mathscr{P}} \in \mathbb{Z}$ and $n_{\mathscr{P}} = 0$ for all but a finite number of places $\mathscr{P}$. The places in $\Sigma(F)$ are called **prime divisors** and we denote the coefficients of the prime divisors of D as $\text{ord}_P(D) = n_{\mathscr{P}}$. The **support** of D, denoted $\text{supp}(D)$, is the set of places $\mathscr{P} \in \Sigma(F)$ with non-zero coefficients $\text{ord}_{\mathscr{P}}(D)$. If $\text{ord}_{\mathscr{P}}(D) \geq 0$ for all $\mathscr{P} \in \text{supp}(D)$, then D is **effective**, denoted $D \geq 0$. The **degree** of a divisor D is defined as .

$$\deg D = \sum_{\mathscr{P} \in \text{supp}(D)} \text{ord}_{\mathscr{P}}(D) \cdot \deg \mathscr{P}.$$

The set of divisors on F forms a free Abelian group $\text{Div}(F)$ under addition.

For any place $\mathscr{P} \in \Sigma(F)$, let $v_{\mathscr{P}}$ denote its valuation. Every non-zero $f \in F$ has an associated divisor given as follows:

$$\text{div}(f) = \sum_{\mathscr{P} \in \Sigma(F)} v_{\mathscr{P}}(f) \cdot \mathscr{P}.$$

The divisors that can be written in the form $\text{div}(f)$, for some non-zero $f \in F$, are the **principal divisors**.

Let $f \in F$ and $\mathscr{P} \in \Sigma(F)$ with corresponding discrete valuation $v_{\mathscr{P}}$. The place

$\mathscr{P}$ is a **zero** of order m at f if and only if $v_{\mathscr{P}}(f) = m > 0$. The place $\mathscr{P}$ is a **pole** of order m at f if and only if $v_{\mathscr{P}}(f) = -m < 0$ [Sti93, p. 7].

Let F' be an algebraic extension of the function field F, both defined over the same constant field K. The **conorm** of a divisor is given by the following homomorphism:

$$\mathrm{Con}_{F'/F} : \begin{cases} \mathrm{Div}(F) & \to & \mathrm{Div}(F') \\ D & \mapsto & \displaystyle\sum_{\mathscr{P}\in \mathrm{supp}(D)} \sum_{\substack{\mathscr{P}'\in\Sigma(F') \\ \mathscr{P}'|\mathscr{P}}} \mathrm{ord}_{\mathscr{P}}(D)\cdot e(\mathscr{P}'|\mathscr{P})\cdot \mathscr{P}' . \end{cases}$$

For any divisor $D \in \mathrm{Div}(F)$, the degree of the divisor under the conorm map is related by

$$\deg \mathrm{Con}_{F'/F}(D) = [F' : F] \cdot \deg D .$$

If D is a principal divisor, then $\mathrm{Con}_{F'/F}(D) = D$, the result is a principal divisor in $\mathrm{Div}(F')$ [Sti93, pp. 63–66].

2.2.4 Differentials and Canonical Divisors

An **adèle** of F is a sequence $(\alpha_{\mathscr{P}})_{\mathscr{P}\in\Sigma(F)}$ such that $\alpha_{\mathscr{P}} \in \mathcal{O}_{\mathscr{P}}$ for all but a finite number of $\mathscr{P} \in \Sigma(F)$. Then if $v_{\mathscr{P}}$ is the valuation of $\mathcal{O}_{\mathscr{P}}$, we have that $v_{\mathscr{P}}(\alpha_{\mathscr{P}}) \geq 0$ for all but a finite number of $\mathscr{P} \in \Sigma(F)$. To simplify notation, we let $\alpha = (\alpha_{\mathscr{P}})_{\mathscr{P}\in\Sigma(F)}$ and $v_{\mathscr{P}}(\alpha) = v_{\mathscr{P}}(\alpha_{\mathscr{P}})$. Then an adèle α is an element of the following set:

$$\mathcal{A}_F = \coprod_{\mathscr{P}\in\Sigma(F)} \widehat{F}_{\mathscr{P}} ,$$

where $\widehat{F}_{\mathscr{P}}$ denotes the completion of F with respect to $\mathscr{P}$.[4] Since F can be embedded in $\mathcal{A}_F$, this restricted direct product results in $\mathcal{A}_F$ being a vector space over K, called the **adèle space** of F [Neu99, pp. 357–358].

Adèles have addition and multiplication operations defined componentwise over the sequence. We say that two adèles $\alpha_1, \alpha_2 \in \mathcal{A}_F$ are **congruent** modulo a divisor $D \in \mathrm{Div}(F)$ if

$$v_{\mathscr{P}}(\alpha_1) - v_{\mathscr{P}}(\alpha_2) = v_{\mathscr{P}}(\alpha_1 - \alpha_2) \geq \mathrm{ord}_{\mathscr{P}}(D) \ ,$$

for every place $\mathscr{P} \in \Sigma(F)$. This congruence is denoted in the natural way: $\alpha_1 \equiv \alpha_2 \pmod{D}$. Then for any divisor $D \in \mathrm{Div}(F)$ we define the following subspace:

$$\mathcal{A}_F(D) = \left\{ \alpha \in \mathcal{A}_F \;\middle|\; \alpha \equiv 0 \pmod{D} \right\} .$$

A **differential** of F is a K-linear map $\omega : \mathcal{A}_F \to K$ such that ω vanishes on $\mathcal{A}_F(-D)+F$ for some divisor $D \in \mathrm{Div}(F)$. Note that $\mathcal{A}_F(-D) + F$ is the space containing elements of the form $\alpha + f = (\alpha_{\mathscr{P}} + f)_{\mathscr{P} \in \Sigma(F)}$ for $f \in F$ and $\alpha \in \mathcal{A}_F$ satisfying $v_{\mathscr{P}}(\alpha) \geq -\mathrm{ord}_{\mathscr{P}}(D)$ for every $\mathscr{P} \in \Sigma(F)$. The set of all differentials of F, denoted $\Omega_{F/K}$, is called the **module of differentials** of F [Che51, pp. 25–30].

Each non-zero differential $\omega \in \Omega_{F/K}$ has a unique divisor $W \in \mathrm{Div}(F)$ satisfying the following properties:

i) ω vanishes on $\mathcal{A}_F(-W) + F$.

ii) For any $D \in \mathrm{Div}(F)$ if ω vanishes on $\mathcal{A}(-D) + F$, then $W - D \geq 0$.

[4] The "completion of F with respect to $\mathscr{P}$" is the superset $\widehat{F}_{\mathscr{P}} \supseteq F$ such that every Cauchy sequence in F is convergent in $\widehat{F}_{\mathscr{P}}$ with respect to the valuation $v_{\mathscr{P}}$ [Neu99, p. 123].

This divisor $W = \mathrm{div}(\omega)$ is called a **canonical divisor** of F [Sti93, p. 27].

2.2.5 Riemann-Roch and the Genus

The Riemann-Roch theorem as given earler for curves, also applies to function fields. Let F be an algebraic function field over a field K. For each divisor $D \in \mathrm{Div}(F)$, the **Riemann-Roch space** is the vector space $\mathscr{L}(D)$ over K given by

$$\mathscr{L}(D) = \left\{ f \in F \setminus \{0\} \;\middle|\; \mathrm{div}(f) + D \geq 0 \right\} \cup \{0\} .$$

The dimension of the Riemann-Roch space is denoted $\ell(D) = \dim_K \mathscr{L}(D)$ [Sti93, pp. 16–17].

The **Riemann-Roch theorem** is of fundamental importance to algebraic function fields [Sti93, pp. 28–29]:

Theorem 2.8 (Riemann, 1857; Roch, 1865). *There exists an integer $g \geq 0$ satisfying*

i) $\ell(D) - \ell(W - D) = \deg(D) - g + 1$ *for any divisor* $D \in \mathrm{Div}(F)$ *and canonical divisor* $W \in \mathrm{Div}(F)$.

ii) $\ell(D) = \deg(D) - g + 1$ *for any divisor* $D \in \mathrm{Div}(F)$ *with* $\deg D > 2g - 2$.

The value g is called the **genus** of the function field F. If F has genus $g = 0$ and there exists a divisor of degree one, then F is precisely the rational function field $K(x)$ for some x transcendental over K. If F has genus $g = 1$ and there exists a divisor of degree one, then F is called an **elliptic function field**. Note that if K is algebraically closed or a finite field (the cases in which we are interested), then there

always exists a divisor of degree one. We consider function fields of genus $g > 1$ in Section 2.2.7, but first we introduce another formula to compute the genus.

2.2.6 The Different and the Hurwitz Formula

Let F' denote an algebraic function field over K that is a finite algebraic extension of the rational function field $F = K(x)$. We define a divisor called the **different** of F'/F as

$$\mathrm{Diff}(F'/F) = \sum_{\mathscr{P} \in \Sigma(F)} \sum_{\substack{\mathscr{P}' \in \Sigma(F') \\ \mathscr{P}' | \mathscr{P}}} d(\mathscr{P}' | \mathscr{P}) \cdot \mathscr{P}' , \tag{2.9}$$

where $d(\mathscr{P}'|\mathscr{P})$ is called the **different exponent** of $\mathscr{P}'$ over $\mathscr{P}$ and is given in the **Dedekind different theorem** as the following [Sti93, pp. 82–83,89]:

Theorem 2.9 (Dedekind). *Let F' be an algebraic function field, a finite separable extension of F with the same constant field K. Then for any place $\mathscr{P}' \in \Sigma(F')$ lying above a place $\mathscr{P} \in \Sigma(F)$ the different exponent satisfies*

$$d(\mathscr{P}'|\mathscr{P}) \geq e(\mathscr{P}'|\mathscr{P}) - 1 .$$

If F' is a tame extension of F, then the above inequality is an equality; otherwise, the inequality is strict.

This leads to the **Hurwitz formula** which allows one to compute the genus of a function field [Sti93, p. 88].

Theorem 2.10 (Hurwitz, 1891). *Let F' be an algebraic function field, a finite separable extension of F with the same constant field K. Then the genus g' of F' satisfies*

$$2g' - 2 = [F' : F](2g - 2) + \deg \mathrm{Diff}(F'/F) .$$

From Dedekind's theorem and the definition of the different in Equation (2.9) we get the following corollary to the Hurwitz formula [Sti93, p. 95]:

Corollary 2.11. *Let F' be an algebraic function field, a finite separable extension of F with the same constant field K. Then the genus g' of F' satisfies*

$$2g' - 2 \geq [F' : F](2g - 2) + \sum_{\mathscr{P} \in \Sigma(F)} \sum_{\substack{\mathscr{P}' \in \Sigma(F') \\ \mathscr{P}' | \mathscr{P}}} \big(e(\mathscr{P}'|\mathscr{P}) - 1\big) \deg(\mathscr{P}') .$$

If F' is a tame extension of F, then the above inequality is an equality; otherwise, the inequality is strict.

2.2.7 Hyperelliptic Function Fields

Let K be a perfect field and $K(x)$ the rational function field over K, where x is transcendental over K. A **hyperelliptic function field** F is a separable degree 2 extension of $K(x)$. Because it is a degree 2 extension, F is sometimes called a **quadratic function field**. Moreover, we have that $F = K(x, y)$ for some $y \in F$ algebraic over $K(x)$; that is, y is a root of some irreducible, separable, quadratic polynomial $\Psi(T)$ in the polynomial ring $K(x)[T]$ for an arbitrary indeterminate T. If $\mathrm{char}\, K \neq 2$, we can define $F = K(x, y)$ by a polynomial $\Psi(T) = T^2 - f(x)$ with $f \in K[x]$ squarefree [Sti93, pp. 108, 113, 193–194].

We continue to assume $\operatorname{char} K \neq 2$. Since we have $[F : K(x)] = 2$, Corollary 2.11 of the Hurwitz formula simplifies to

$$2g + 2 = \sum_{\mathscr{P} \in \Sigma(K(x))} \sum_{\substack{\mathscr{P}' \in \Sigma(F) \\ \mathscr{P}' | \mathscr{P}}} \left(e(\mathscr{P}'|\mathscr{P}) - 1\right) \deg(\mathscr{P}') \,,$$

where each ramification index $e(\mathscr{P}'|\mathscr{P}) \leq 2$. Therefore, there are at most $2g + 2$ ramified places in F over $K(x)$. However, if $\mathscr{P}_\infty \in \Sigma(K(x))$ is ramified in F, then there are only $2g + 1$ ramified finite places. In the case that $\mathscr{P}_\infty$ is ramified in F, we say that F is **imaginary**. Otherwise, if there are two distinct places lying above $\mathscr{P}_\infty$ in F, then there are $2g + 2$ ramified finite places and we say that F is **real**. There is a degenerate case when $\mathscr{P}_\infty$ remains prime (and unramified) in F and $\deg \mathscr{P}'_\infty = 2$ for $\mathscr{P}'_\infty | \mathscr{P}_\infty$; in this case F is called **unusual** [PR99, §§3–4].

If $\operatorname{char} K = 2$, we define $F = K(x, y)$ by a polynomial $\Psi(T) = T^2 + h(x)T - f(x)$ such that y is a root of $\Psi(T)$ with $f, h \in K[x]$ and h non-zero. The function field F is no longer tame in this case, so we must use the Hurwitz formula (Theorem 2.10) which simplifies to

$$2g + 2 = \deg \operatorname{Diff}(F'/K(x)) \,.$$

Since every ramified place is wildly ramified, Dedekind's different theorem (Theorem 2.9)) gives that each different exponent $d(\mathscr{P}'|\mathscr{P}) \geq 2$. Therefore, from the definition of the different in Equation (2.9), the number r of ramified places in F over $K(x)$ must be in the range $1 \leq r \leq g + 1$ [Sti93, p. 194]. Now, if $\mathscr{P}_\infty \in \Sigma(K(x))$ is ramified in F, then there are at most g ramified finite places and we call F **imagi-**

nary. Otherwise, if there are two distinct places lying above $\mathscr{P}_\infty$ in F and there are $g+1$ ramified finite places, then we say that F is **real**. We still have the degenerate case where F is **unusual** in which $\mathscr{P}_\infty$ remains prime (and unramified) in F and $\deg \mathscr{P}'_\infty = 2$ for $\mathscr{P}'_\infty | \mathscr{P}_\infty$.

In the rest of this thesis we ignore the case that F is unusual. As mentioned in Section 2.1.8, an unusual quadratic function field F with constant field K is real quadratic when considered over a degree 2 constant field extension L of K [PR99, §1]. We focus on real quadratic function fields after briefly describing the relationship between real and imaginary quadratic function fields in the beginning of the next chapter.

Chapter 3

Real Quadratic Function Fields

As was shown in the previous chapter, the concepts of divisors, the Riemann-Roch theorem and the Hurwitz formula apply to both algebraic curves and algebraic function fields. In fact, there is a one-to-one correspondence between points and places in the sense that both are prime divisors. Therefore, when one considers quadratic function fields, the theory from algebraic geometry and from purely algebraic structures coalesce into a unified theory. An imaginary quadratic function field is the function field of an imaginary hyperelliptic curve. Similarly, a real quadratic function field is the function field of a real hyperelliptic curve. This is summarized in the following section.

In Section 3.2 we focus on algorithms in the infrastructure of a real quadratic function field. Section 3.3 concludes with a discussion of efficient arithmetic for operations in the infrastructure.

3.1 Coalescence in Quadratic Function Fields

In this section we summarize results from the previous chapter. Consider a hyperelliptic function field F of genus g defined over a finite field $K = \mathbb{F}_q$, where F is an quadratic extension of the rational function field $K(x)$. Then F is equal to the function field $K(C)$ of a hyperelliptic curve C given by

$$C : y^2 + h(x)y = f(x) ,$$

where $f, h \in K[x]$. Also, C must satisfy the condition that no point $P = (a, b) \in C(\overline{K})$ satisfies both $2b + h(a) = 0$ and $f'(a) - h'(a)b = 0$. We call F an **imaginary quadratic function field** if we can put Equation (3.1) into a birationally equivalent canonical form such that

i) If $\operatorname{char} K \neq 2$, then $h = 0$, f is monic and squarefree with $\deg f = 2g + 1$;

ii) If $\operatorname{char} K = 2$, then $h \neq 0$, $\deg h \leq g$, f is monic with $\deg f = 2g + 1$.

We call F a **real quadratic function field** if we can put Equation (3.1) into a birationally equivalent canonical form such that

i) If $\operatorname{char} K \neq 2$, then $h = 0$, f is squarefree with $\deg f = 2g + 2$ and $\operatorname{sgn}(f) = e^2$ for some $e \in K^*$;

ii) If $\operatorname{char} K = 2$, then $h \neq 0$, h is monic, $\deg h = g + 1$ and either

   (a) $\deg f \leq 2g + 1$, or

   (b) $\deg f = 2g + 2$ and $\operatorname{sgn}(f) = e^2 + e$ for some $e \in K^*$.

We can also determine F to be an imaginary quadratic function field if either of the following properties are satisfied:

i) $C \cap H_\infty = \{P_\infty\}$ where P_∞ is a Weierstrass point of C;

ii) $\mathscr{P}_\infty \in \Sigma(K(x))$ is ramified in F.

Similarly, F is a real quadratic function field if the following properties are satisfied:

i) $C \cap H_\infty = \{P_\infty^1, P_\infty^2\}$ where P_∞^1 and P_∞^2 are not Weierstrass points of C;

ii) $\mathscr{P}_\infty \in \Sigma(K(x))$ splits into two distinct places in F.

We recall from Section 2.1.8 that C has a special map called the **hyperelliptic involution** defined by $\iota : y \mapsto -y - h(x)$.

The terms "imaginary quadratic" and "real quadratic" come from the fact that imaginary quadratic function fields have similarities with imaginary quadratic number fields and likewise real quadratic function fields are similar to real quadratic number fields. A **quadratic number field** is a degree 2 field extension of the rational field $\mathbb{Q}$. It turns out that any quadratic number field can be expressed as $\mathbb{Q}(\sqrt{d})$, where d is a squarefree integer. In the case that $d > 0$, we have that $\mathbb{Q}(\sqrt{d})$ is a subfield of $\mathbb{R}$ and consequently we call $\mathbb{Q}(\sqrt{d})$ a **real quadratic number field**. When $d < 0$ we call $\mathbb{Q}(\sqrt{d})$ an **imaginary quadratic number field** since it is a subfield of $\mathbb{C}$ but not of $\mathbb{R}$ [HW79, pp. 204, 208].

The coalescence in the theory extends further. In this section we will show how divisors (in either function fields or on curves) correspond to ideals and that arithmetic on one is equivalent to performing arithmetic on the other. This is most straightforward in the case of imaginary quadratic function fields; however, this correspondence includes real quadratic function fields at the cost of some extra work.

3.1.1 Divisor Class Group

Let F be a hyperelliptic function field corresponding to a hyperelliptic curve C over a field K. Two divisors $D, D' \in \mathrm{Div}(F)$ are **equivalent**, denoted $D \sim D'$, if $D - D'$ is a principal divisor. Let $\mathrm{Div}^d(F)$ denote the set of divisors of degree d and let $\mathrm{Prin}(F)$ be the set of principal divisors. Then we have the following sequence of subgroups: $\mathrm{Div}(F) \supset \mathrm{Div}^0(F) \supset \mathrm{Prin}(F)$. The **divisor class group** of F is the quotient group

$$\mathrm{Cl}(F) = \mathrm{Div}(F)/\mathrm{Prin}(F)\ .^1$$

Elements of $\mathrm{Cl}(F)$ are called **divisor classes**. The divisor class group $\mathrm{Cl}(F)$ is a finite Abelian group, giving us the following exact sequence:

$$0 \longrightarrow \mathrm{Prin}(F) \longrightarrow \mathrm{Div}(F) \longrightarrow \mathrm{Cl}(F) \longrightarrow 0\ .$$

Denote by $\mathrm{Cl}^0(F)$ the kernel of the degree map $\mathrm{Cl}(F) \to \mathbb{Z}$. The set $\mathrm{Cl}^0(F)$ consists of the equivalence classes of degree zero divisors and is a subgroup of $\mathrm{Cl}(F)$ [Har77, pp. 131, 139–140]. Therefore, we have

$$0 \longrightarrow \mathrm{Cl}^0(F) \longrightarrow \mathrm{Cl}(F) \longrightarrow \mathbb{Z} \longrightarrow 0\ .$$

The degree-zero divisor class group is isomorphic to the **Jacobian** of the curve C.[2] Therefore, arithmetic with divisor classes in $\mathrm{Cl}^0(F)$ can be said to be performed on

[1] The divisor class group $\mathrm{Cl}(F)$ is sometimes denoted $\mathrm{Pic}(C)$ as there is an isomorphism between $\mathrm{Cl}(F)$ and the Picard group of C (*cf.* Hartshorne [Har77, pp. 143–145]).

[2] The Jacobian is an Abelian variety that is isomorphic to $\mathrm{Cl}^0(F)$ (*cf.* Hartshorne [Har77, p. 140]). If C is a curve of genus g over the complex field $\mathbb{C}$, then the Jacobian of C is a torus of dimension g. An explicit definition of the Jacobian is beyond the scope of this work (*cf.* Birkenhake and Lange [BL04, pp. 316–321]).

the Jacobian.

Let h_F denote the cardinality of $\mathrm{Cl}^0(F)$, called the **class number** of $\mathrm{Cl}^0(F)$. An upper and lower bound on h_F is due Hasse and Weil as a consequence of their proof of the Riemann hypothesis in function fields [Ros02, p. 55].

Theorem 3.1 (Hasse, 1935; Weil, 1948). *If F is a hyperelliptic function field of genus g over a finite field $K = \mathbb{F}_q$, the class number h_F of $\mathrm{Cl}^0(F)$ is bounded by*

$$(\sqrt{q} - 1)^{2g} \leq h_F \leq (\sqrt{q} + 1)^{2g}.$$

Artin gave an upper bound on the class number which differentiates between the cases where F is imaginary or real quadratic [Art21, p. 236].

Theorem 3.2 (Artin, 1921). *If F is a hyperelliptic function field of genus g over a finite field $K = \mathbb{F}_q$, the class number h_F of $\mathrm{Cl}^0(F)$ is bounded by*

$$h_F \leq \begin{cases} q^g \deg f & \text{if } F \text{ is imaginary quadratic} \\ 2q^{g-1}(\deg f - 1)^2 & \text{if } F \text{ is real quadratic}. \end{cases}$$

The next subsection shows how we can find a representative for each divisor class of $\mathrm{Cl}^0(F)$.

3.1.2 Reduced Divisors

Let $F = K(C)$ be the function field of genus g corresponding to the hyperelliptic curve $C : y^2 + h(x)y = f(x)$. Recall that F is a quadratic extension of the rational function field $K(x)$. Let $\mathcal{P}$ denote a prime divisor (either a point $P \in C$ or a place

$\mathscr{P} \in \Sigma(F)$). We denote the **prime divisor at infinity** in $K(x)$ as $\mathcal{P}_\infty$, representing either the place at infinity $\mathscr{P}_\infty$ of F or the point at infinity P_∞ on C. Let S denote the set of prime divisors at infinity in F. If F is imaginary quadratic, then $S = \{\mathcal{P}'_\infty\}$; otherwise, if F is real quadratic, then $S = \{\mathcal{P}^1_\infty, \mathcal{P}^2_\infty\}$. A divisor $D \in \mathrm{Div}(F)$ is finite if its support is disjoint from S.

For any divisor $D \in \mathrm{Div}^0(F)$, there exists a divisor $D' \sim D$ such that D' is the difference of two divisors $D' = D_S - \widetilde{D_S}$, where D_S is effective and $\widetilde{D_S}$ is balanced at infinity as

$$\widetilde{D_S} = \begin{cases} g\mathcal{P}_\infty, & \text{if } F \text{ is imaginary quadratic} \\ \frac{g}{2}(\mathcal{P}^1_\infty + \mathcal{P}^2_\infty), & \text{if } F \text{ is real quadratic and } g \text{ is even} \\ \frac{g+1}{2}\mathcal{P}^1_\infty + \frac{g-1}{2}\mathcal{P}^2_\infty, & \text{if } F \text{ is real quadratic and } g \text{ is odd}. \end{cases}$$

We call D' **semi-reduced** if every two non-equal prime divisors $\mathcal{P}_i, \mathcal{P}_j \in \mathrm{supp}(D_S)$ satisfy $\mathcal{P}_i \neq \overline{\mathcal{P}}_j$, where $\overline{\mathcal{P}}_j$ denotes the conjugate of the prime divisor $\mathcal{P}_j$ [GHMM08, §2].

If F is imaginary quadratic, a semi-reduced divisor $D = D_S - \widetilde{D_S}$ can be uniquely represented as a pair of polynomials $a, b \in K[x]$ given by

$$a(x) = \prod_{P=(a_i,b_i)\in \mathrm{supp}(D_S)\setminus S} (x - a_i)^{\mathrm{ord}_P(D_S)}$$
$$b(a_i) = b_i \quad \text{for every} \quad (a_i, b_i) \in \mathrm{supp}(D_S)$$

with the additional conditions that a and b satisfy $\deg b < \deg a$ and $a|(f - bh - b^2)$. This polynomial representation of a semi-reduced divisor is called **Mumford**

representation, and we denote such a divisor as $\mathrm{div}(a, b)$ [GHMM08, §2].

A semi-reduced divisor $D = \mathrm{div}(a, b)$ is called **reduced** if $\deg a \leq g$, where g is the genus of F. If F is imaginary quadratic, then each class of $\mathrm{Cl}^0(F)$ contains exactly one reduced divisor. If F is real quadratic, then each class of $\mathrm{Cl}^0(F)$ has a representative of the form $D_S - \widetilde{D_S}$, where we write $D_S = D'_S + n_S\mathcal{P}^1_\infty + m_S\mathcal{P}^2_\infty$ for a finite reduced divisor D'_S and $n_S, m_S \in \mathbb{Z}_{\leq 0}$. Then we can uniquely denote each equivalence class in the real quadratic case with $\mathrm{div}(a_S, b_S)$, the Mumford representation of D_S, along with n_S [GHMM08, §§2, 4].

Algorithms for performing operations on divisors in Mumford representation are provided by Galbraith, Harrison and Mireles Morales [GHMM08]. However, in this thesis we will work with ideals instead of divisors.

3.1.3 Fractional and Reduced Ideals

Let $F = K(C)$ be the function field of a hyperelliptic curve $C : y^2 + h(x)y = f(x)$ and let $\mathcal{O}$ denote the coordinate ring $K[C]$. In general, if $F = K(x, y)$ is an extension of the rational function field $K(x)$, then $\mathcal{O}$ is the integral closure of $K[x]$ in F.[3] In the case of hyperelliptic function fields, $\mathcal{O}$ is called the (maximal) **quadratic order** of F. Note that the field of quotients $\mathrm{Quot}(\mathcal{O}) = F$. Any element $\alpha \in F$ can be written uniquely as $\alpha = (a + by)/d$ for some $a, b, d \in K[x]$, where $d \neq 0$ and $\gcd(a, b, d) = 1$. Note that $\alpha \in \mathcal{O}$ if and only if $d \in K^*$.

A **fractional ideal** of $\mathcal{O}$ is an $\mathcal{O}$-submodule $\mathfrak{a}$ of F such that there exists a non-zero "denominator" $\beta \in \mathcal{O}$ where $\mathfrak{a} \subseteq (1/\beta)\mathcal{O}$. To avoid confusion, sometimes ordinary ideals are called **integral ideals** since they are the special case where $\beta \in$

[3] For a subring S of an integral domain R, the "integral closure of S in R" is the set of all elements $a \in R$ satisfying $f(a) = 0$ for some monic polynomial $f \in S[x]$ [ZS75, V. I, pp. 254–256].

K^*. A fractional $\mathcal{O}$-ideal $\mathfrak{a}$ is contained in $\mathcal{O}$ if and only if it is integral [ZS75, V. I, p. 271].

Consider a rank-2 free $K[x]$-submodule of F given by $\alpha K[x]+\beta K[x]$ for $\alpha, \beta \in F$. We denote this module by its basis $\{\alpha, \beta\}$. Note that two bases $\{\alpha, \beta\}$ and $\{\gamma, \delta\}$ represent the same module if and only if

$$\begin{pmatrix} \gamma \\ \delta \end{pmatrix} = X \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \qquad \text{for some} \quad X \in \mathrm{GL}_2(K[x]) \,, \tag{3.1}$$

where $\mathrm{GL}_2(K[x])$ is the **general linear group** of degree 2, *i.e.* the group of all 2×2 matrices with entries in $K[x]$ and determinant in K^* [WW87, §3].

A fractional ideal $\mathfrak{a} \subseteq F$ can be written as a rank-2 free $K[x]$-submodule of F in the form

$$\mathfrak{a} = s\Big(aK[x] + (b+y)K[x]\Big) \,, \tag{3.2}$$

where $s \in K(x)$ with the denominator of s monic, and $a, b \in K[x]$ satisfy the conditions that a is monic and $a|(f + bh - b^2)$. We can represent $\mathfrak{a}$ with the basis $\{sa, s(b+y)\}$. If $s \in K[x]$, then $\mathfrak{a}$ is an integral ideal. An integral ideal $\mathfrak{a} \subseteq \mathcal{O}$ is represented in the form of Equation (3.2) where $s, a, b \in K[x]$ such that both s and a are monic with $a|(f + bh - b^2)$. If we have $\deg b < \deg a$ by taking b modulo a, then the basis is said to be **adapted** and is unique for the ideal $\mathfrak{a}$. We write an ideal with adapted basis $\{sa, s(b+y)\}$ in **standard represention** as a polynomial triple (s, a, b) [SW99, §3A], [Zuc97a, §6].

An integral $\mathcal{O}$-ideal $\mathfrak{a}$ is called **primitive** if it cannot be written as $\mathfrak{a} = m\mathfrak{b}$ with

$\mathfrak{b}$ an integral $\mathcal{O}$-ideal and a multiplier $m \in K[x] \setminus K$. If $\mathfrak{a}$ is given in standard representation as (s, a, b), then $\mathfrak{a}$ is primitive if and only if $s = 1$, in which case we drop s from the standard representation and write $\mathfrak{a} = (a, b)$.

A **principal fractional ideal** is a fractional ideal such that $\mathfrak{a} = (\alpha/\beta)\mathcal{O}$, where $\beta \neq 0$ is the denominator with $\alpha, \beta \in \mathcal{O}$. Then the value $\gamma = \alpha/\beta = (a + by)/d \in F \setminus \{0\}$ is said to **generate** $\mathfrak{a}$ [ZS75, V. I, p. 271]. Given polynomials $a, b, d \in K[x]$ with $d \neq 0$ and a, b not both zero, the standard representation of the principal fractional ideal generated by $\gamma = (a+by)/d \in F$ can be computed using Algorithm 3.3 [Sch96, §3]. The opposite operation of finding the generator of an ideal given its standard representation is a computationally difficult problem (see Section 4.1).

Algorithm 3.3 (Principal ideal standard representation). *Given $\gamma \in F$, find the standard representation of the principal fractional $\mathcal{O}$-ideal generated by γ.*

Input: A non-zero $\gamma = (a + by)/d \in F$ with $a, b, d \in K[x]$, and $C : y^2 + h(x)y = f(x)$ for $F = K(C)$.

Output: The $\mathcal{O}$-ideal $\mathfrak{a} = (s_\mathfrak{a}, a_\mathfrak{a}, b_\mathfrak{a}) = (\gamma)\mathcal{O}$.

1: $s_\mathfrak{a}, u, v \leftarrow \operatorname{xgcd}(b, a + bh)$ ▷ Compute u, v from extended Euclidean alg.

2: $a_\mathfrak{a} \leftarrow \big(a^2 - b^2 f + abh\big)/s_\mathfrak{a}^2$

3: $b_\mathfrak{a} \leftarrow \big(ua + vbf\big)/s_\mathfrak{a} \pmod{a_\mathfrak{a}}$

4: $s_\mathfrak{a} \leftarrow s_\mathfrak{a}/(\operatorname{sgn}(s_\mathfrak{a})d)$ ▷ Note that $s_\mathfrak{a} \in K(x)$

An ideal in adapted basis is said to be **reduced** if $\deg a \leq g$, the genus of F [JSS07b, §4]. In Section 3.2.1 we will introduce an algorithm to perform "reduction" on a primitive ideal to obtain a reduced ideal.

3.1.4 Multiplying and Inverting Fractional Ideals

Fractional ideals admit a **multiplication** operation $\mathfrak{a} \cdot \mathfrak{b} \subseteq (\beta_\mathfrak{a}\beta_\mathfrak{b})^{-1}\mathcal{O}$ for $\mathcal{O}$-ideals $\mathfrak{a}$ and $\mathfrak{b}$ with denominators $\beta_\mathfrak{a}, \beta_\mathfrak{b} \in \mathcal{O}$, respectively. The identity of this operation is the ring $\mathcal{O}$, itself an ideal, denoted in standard representation as $\mathcal{O} = (1, 0)$. Algorithm 3.4 presents the computations necessary for multiplication, following an algorithm of Cantor [Can87] (with optimizations discussed therein) and generalizations by Koblitz [Kob89]. Note that even if $\mathfrak{a}$ and $\mathfrak{b}$ are primitive, the product is not necessarily primitive. When $\mathfrak{a} = \mathfrak{b}$, squaring an ideal simplifies as shown in Algorthm 3.5.

Algorithm 3.4 (Ideal multiplication). *Multiply two fractional $\mathcal{O}$-ideals in standard representation. All of the operations are performed in the ring $K[x]$.*

Input: Two $\mathcal{O}$-ideals $\mathfrak{a} = (s_\mathfrak{a}, a_\mathfrak{a}, b_\mathfrak{a})$ and $\mathfrak{b} = (s_\mathfrak{b}, a_\mathfrak{b}, b_\mathfrak{b})$, where $\mathcal{O} = K[C]$ for the hyperelliptic curve $C : y^2 + h(x)y = f(x)$ over K.

Output: An $\mathcal{O}$-ideal $\mathfrak{c} = (s_\mathfrak{c}, a_\mathfrak{c}, b_\mathfrak{c}) = \mathfrak{a}\mathfrak{b}$.

1: $(d_1, u_1, v_1) \leftarrow \text{xgcd}(a_\mathfrak{a}, a_\mathfrak{b})$ ▷ Compute u_1, v_1 from extended Euclidean alg.

2: $s_\mathfrak{c} \leftarrow s_\mathfrak{a}s_\mathfrak{b}$, $a_\mathfrak{c} \leftarrow a_\mathfrak{a}a_\mathfrak{b}$

3: $b_\mathfrak{c} \leftarrow u_1 a_\mathfrak{a} b_\mathfrak{b} + v_1 a_\mathfrak{b} b_\mathfrak{a}$

4: **if** $d_1 \neq 1$ **then**

5: $\quad (d_2, u_2, v_2) \leftarrow \text{xgcd}(d_1, b_\mathfrak{a} + b_\mathfrak{b} + h)$ ▷ Extended Euclidean alg.

6: $\quad s_\mathfrak{c} \leftarrow s_\mathfrak{c} d_2$

7: $\quad a_\mathfrak{c} \leftarrow a_\mathfrak{c}/d_2^2$

8: $\quad b_\mathfrak{c} \leftarrow \big(u_2 b_\mathfrak{c} + v_2(b_\mathfrak{a}b_\mathfrak{b} + f)\big)/d_2$

9: $b_\mathfrak{c} \leftarrow b_\mathfrak{c} \pmod{a_\mathfrak{c}}$

Algorithm 3.5 (Ideal squaring). *Squaring a fractional $\mathcal{O}$-ideal in standard representation. All of the operations are performed in the ring $K[x]$.*

Input: An $\mathcal{O}$-ideal $\mathfrak{a} = (s_\mathfrak{a}, a_\mathfrak{a}, b_\mathfrak{a})$, where $\mathcal{O} = K[C]$ for the hyperelliptic curve C : $y^2 + h(x)y = f(x)$ over K.

Output: An $\mathcal{O}$-ideal $\mathfrak{c} = (s_\mathfrak{c}, a_\mathfrak{c}, b_\mathfrak{c}) = \mathfrak{a}^2$.

1: $(d, u, v) \leftarrow \text{xgcd}(a_\mathfrak{a}, 2b_\mathfrak{a} + h)$ ▷ Compute u, v from extended Euclidean alg.

2: $s_\mathfrak{c} \leftarrow s_\mathfrak{a}^2 d$

3: $a_\mathfrak{c} \leftarrow (a_\mathfrak{a}/d)^2$

4: $b_\mathfrak{c} \leftarrow (u a_\mathfrak{a} b_\mathfrak{a} + v(b_\mathfrak{a}^2 + f))/d \pmod{a_\mathfrak{c}}$

Let $L(n) = \log n \log\log n$. Multiplying or dividing degree n polynomials with remainder both require $O(nL(n))$ operations in K [AHU74, pp. 286–292]. We can compute the GCD of two degree n polynomials in $O(nL(n)\log n)$ operations in K [AHU74, pp. 303–310]. Therefore, if $a_\mathfrak{a}$ and $a_\mathfrak{b}$ both have degree in $O(n)$, Algorithms 3.4 and 3.5 each run in $O(nL(n)\log n)$. Furthermore, the ideal $\mathfrak{c} = \mathfrak{a}\mathfrak{b}$ will have degree $2n \in O(n)$.

We call a fractional $\mathcal{O}$-ideal $\mathfrak{a} \neq \mathcal{O}$ **prime** if for any $\alpha, \beta \in \mathcal{O}$ such that the product $\alpha\beta$ is in $\mathfrak{a}$, either $\alpha \in \mathfrak{a}$ or $\beta \in \mathfrak{a}$ [ZS75, V. I, p. 149]. A primitive $\mathcal{O}$-ideal $\mathfrak{a} = (a, b)$ in standard representation is prime if and only if a is irreducible in $K[x]$.

Prime ideals have a similar ramification theory as prime divisors (*cf.* Sections 2.1.2 and 2.2.2). We simplify the discussion here for our situation where $\mathcal{O}$ is a quadratic order. Each prime $K[x]$-ideal $\mathfrak{p}$ is principal and generated by a monic irreducible polynomial $p \in K[x]$. Let $\mathfrak{P}$ denote a prime $\mathcal{O}$-ideal. If $\mathfrak{p}\mathcal{O} = \mathfrak{P}\overline{\mathfrak{P}}$, then $\mathfrak{P}$ and $\overline{\mathfrak{P}}$ are said to **lie over** $\mathfrak{p}$ and that $\mathfrak{p}$ **splits** in $\mathcal{O}$. If $\mathfrak{p}\mathcal{O} = \mathfrak{P}^2$, then only $\mathfrak{P}$ lies over $\mathfrak{p}$ and it is said that $\mathfrak{p}$ **ramifies** in $\mathcal{O}$. The other case is if $\mathfrak{p}\mathcal{O} = \mathfrak{P}$, where $\mathfrak{p}$ is called

inert [Neu99, pp. 45–49].

Let $\mathfrak{a} = \{\alpha, \beta\}$ be a fractional $\mathcal{O}$-ideal given by an arbitrary $K[x]$-basis. The **norm** of $\mathfrak{a}$ is defined by $\mathrm{N}(\mathfrak{a})^2(2y+h)^2 = \det(X)^2$, where

$$X = \begin{pmatrix} \alpha & \beta \\ \overline{\alpha} & \overline{\beta} \end{pmatrix},$$

and $\overline{\alpha}$ denotes the **conjugate** of $\alpha \in F$ under the hyperelliptic involution ι. Defined in this way, the norm is independent of the choice of basis. If $\mathfrak{a}$ is given in standard representation as $\mathfrak{a} = (s, a, b)$—recall this represents a $K[x]$-basis $\{sa, s(b+y)\}$—then the norm of $\mathfrak{a}$ is computed as

$$\mathrm{N}(\mathfrak{a}) = as^2 .$$

If $\mathfrak{a}$ is integral, then $\mathrm{N}(\mathfrak{a}) \in K[x]$; if $\mathfrak{a}$ is primitive, then $\mathrm{N}(\mathfrak{a}) = a$. The norm is also completely multiplicative, *i.e.* it satisfies $\mathrm{N}(\mathfrak{a})\,\mathrm{N}(\mathfrak{b}) = \mathrm{N}(\mathfrak{ab})$. Clearly, the norm of the identity $\mathcal{O}$ must be $\mathrm{N}(\mathcal{O}) = 1$ [Ste99, §2].[4]

A fractional $\mathcal{O}$-ideal $\mathfrak{a}$ is **invertible** if there exists a fractional $\mathcal{O}$-ideal $\mathfrak{a}^{-1}$ such that $\mathfrak{a}\mathfrak{a}^{-1} = \mathcal{O}$. This inverse exists if and only if the ideal product $\mathfrak{a}_\mathcal{P} = \mathfrak{a}\mathfrak{m}_\mathcal{P}$ is a prime ideal for every non-zero prime divisor $\mathcal{P} \in \mathrm{Div}(F)$ and corresponding maximal ideal $\mathfrak{m}_\mathcal{P}$ of $\mathcal{O}_\mathcal{P}$. Since $\mathcal{O}$ is a Dedekind domain, all fractional $\mathcal{O}$-ideals are invertible

[4]Many sources in the literature specify that the norm should be made monic. However, since we are assuming that the polynomials in the standard representation come from an adapted basis, both s and a are monic, thus as^2 is always monic.

[Neu99, pp. 74–75]. The inverse of an $\mathcal{O}$-ideal is given by

$$\mathfrak{a}^{-1} = \frac{1}{\mathrm{N}(\mathfrak{a})}\,\overline{\mathfrak{a}}\,, \tag{3.3}$$

where $\overline{\mathfrak{a}}$ denotes the **conjugate** of $\mathfrak{a}$ under the hyperelliptic involution ι. If $\mathfrak{a} = (s, a, b)$ in standard representation, then the conjugate is $\overline{\mathfrak{a}} = (s, a, -b - h)$. Therefore, one can derive from Algorithm 3.4 that $\mathfrak{a}\overline{\mathfrak{a}}/(as^2) = \mathcal{O}$.

3.1.5 Ideal Class Group

Let $\mathcal{O}$ be the integral closure of $K[x]$ in the function field F. Let $\mathrm{Frac}(\mathcal{O})$ denote the set of fractional ideals of $\mathcal{O}$. Since $\mathcal{O}$ is a Dedekind domain, recall that all of the fractional $\mathcal{O}$-ideals are invertible. Therefore, the set $\mathrm{Frac}(\mathcal{O})$ is a group under multiplication. The set $\mathrm{Prin}(\mathcal{O})$ of principal fractional ideals is a subgroup of $\mathrm{Frac}(\mathcal{O})$.

The **ideal class group** is an Abelian group defined to be the quotient group

$$\mathrm{Cl}(\mathcal{O}) = \mathrm{Frac}(\mathcal{O})/\,\mathrm{Prin}(\mathcal{O})\,.$$

Consider the map $\alpha \mapsto (\alpha)\mathcal{O}$ from (non-zero) function field elements to fractional ideals. Then we have the following exact sequence:

$$0 \longrightarrow \mathcal{O}^* \longrightarrow F^* \longrightarrow \mathrm{Frac}(\mathcal{O}) \longrightarrow \mathrm{Cl}(\mathcal{O}) \longrightarrow 0\,,$$

where the class group measures the expansion from $F^* = F \setminus \{0\}$ to the fractional ideals and, similarly, the unit group $\mathcal{O}^*$ measures the contraction in the same map [Neu99, p. 22]. Elements of $\mathrm{Cl}(\mathcal{O})$ are called **ideal classes** of $\mathcal{O}$. Two integral ideals

$\mathfrak{a}, \mathfrak{b} \subseteq \mathcal{O}$ in the same ideal class of $\mathrm{Cl}(\mathcal{O})$ are said to be **equivalent**, denoted $\mathfrak{a} \sim \mathfrak{b}$. We have $\mathfrak{a} \sim \mathfrak{b}$ if and only if there exist non-zero $\alpha, \beta \in F$ such that $(\alpha)\mathfrak{a} = (\beta)\mathfrak{b}$. Then $\mathfrak{a} = (\gamma)\mathfrak{b}$, where $\gamma = \beta/\alpha \in F$ is called the **relative generator** of $\mathfrak{a}$ with respect to $\mathfrak{b}$.[5] Therefore, all principal ideals $\mathfrak{a} = (\gamma)\mathcal{O}$ are contained in one ideal class of $\mathrm{Cl}(\mathcal{O})$, called the **principal ideal class**. The number of ideal classes is the **class number** of $\mathcal{O}$, denoted $h_{\mathcal{O}}$.

If F is an imaginary quadratic function field, then each ideal class of $\mathrm{Cl}(\mathcal{O})$ contains exactly one reduced ideal. Therefore, reduced ideals can be used as representatives of $\mathrm{Cl}(\mathcal{O})$. Artin proved the following important result [Art21, p. 178]:

Theorem 3.6 (Artin, 1921). *If F is an imaginary quadratic function field, then the set of reduced $\mathcal{O}$-ideals is isomorphic to the degree-zero divisor class group $\mathrm{Cl}^0(F)$.*

Thus, we have an equality between the class numbers $h_F = h_{\mathcal{O}}$. However, if F is real quadratic, each ideal class of $\mathrm{Cl}(\mathcal{O})$ may contain multiple reduced ideals.

There exists a **fundamental unit** $\eta \in \mathcal{O}^*$ such that every unit $\varepsilon \in \mathcal{O}^*$ can be written as $\varepsilon = c\eta^m$ for some $c \in K^*$ and $m \in \mathbb{Z}$. The degree of η is called the **regulator** of $\mathcal{O}$, denoted $R_{\mathcal{O}}$ [Sch96, §1]. The regulator appears in the following theorem due to Schmidt [Sch31, p. 32], providing a relation between the divisor class number and the ideal class number that extends to real quadratic function fields:

Theorem 3.7 (Schmidt, 1931). *Let h_F be the class number of the degree-zero divisor class group $\mathrm{Cl}^0(F)$, and let $h_{\mathcal{O}}$ be the class number of the ideal class group $\mathrm{Cl}(\mathcal{O})$.*

[5] The relative generator γ in $\mathfrak{a} = (\gamma)\mathfrak{b}$ can, equivalently, be interpreted as the generator of a principal ideal $\mathfrak{c} = (\gamma)\mathcal{O}$ with $\mathfrak{a}$ equal to the ideal product $\mathfrak{cb}$.

Then for any imaginary or real quadratic function field F, we have the relation

$$h_F = h_{\mathcal{O}} R_{\mathcal{O}} .$$

By Theorem 3.6, the regulator $R_{\mathcal{O}} = 1$ if F is imaginary quadratic. In real quadratic function fields, the ideal class number $h_{\mathcal{O}}$ tends to be very small according to the Cohen-Lenstra heuristics [CL84] that were extended to function fields by Friedman and Washington [FW89] and recently proven in the latter case by Achter [Ach06].

Now we shall show precisely how the reduced ideals in an ideal class of $\mathrm{Cl}(\mathcal{O})$ are related to the divisor classes of $\mathrm{Cl}^0(F)$ when F is a real quadratic function field. Let $\mathfrak{a}$ be a reduced $\mathcal{O}$-ideal and let $\mathcal{R}_{\mathfrak{a}}$ denote the set of reduced $\mathcal{O}$-ideals equivalent to $\mathfrak{a}$. Consider the map given by

$$\Psi : \begin{cases} \mathcal{R}_{\mathfrak{a}} & \to \quad \mathrm{Div}^0(F) \\ \mathfrak{b} = (a_{\mathfrak{b}}, b_{\mathfrak{b}}) & \mapsto \quad \mathrm{div}(a_{\mathfrak{b}}, b_{\mathfrak{b}}) - \deg(a_{\mathfrak{b}})\mathcal{P}_{\infty}^2 . \end{cases}$$

Then the following theorem gives the correspondence between $\mathcal{R}_{\mathfrak{a}}$ and the divisor classes of $\mathrm{Cl}^0(F)$ [MM08, §7]:

Theorem 3.8 (Mireles Morales, 2008). *Let $\mathfrak{a}$ be a reduced $\mathcal{O}$-ideal. Then the reduced ideals in the set $\mathcal{R}_{\mathfrak{a}}$ are in one-to-one correspondence via Ψ with reduced, pairwise-inequivalent divisors that form a subset $\Psi(\mathcal{R}_{\mathfrak{a}}) \subset \mathrm{Cl}^0(F)$.*

In the next section we show how one can perform operations in $\mathcal{R}_{\mathfrak{a}}$ via the reduced ideals. The map Ψ respects these operations.

3.2 The Infrastructure of a Real Quadratic Function Field

Given a fixed reduced $\mathcal{O}$-ideal $\mathfrak{a}$, one can step through the ideal class of $\mathfrak{a}$ to find equivalent reduced ideals in $\mathcal{R}_\mathfrak{a}$ using an internal structure called the **infrastructure**. This term was introduced by Shanks [Sha72b] in the case of real quadratic number fields. The **infrastructure** of a real quadratic function field refers to the internal structure of the set of reduced ideals in an equivalence class of $\mathrm{Cl}(\mathcal{O})$.

The infrastructure provides two closed operations called "baby steps" and "giant steps," but we will see that these operations do not endow a complete group structure. The algorithm for performing the baby step operation uses an approach based on continued fraction expansions, while giant steps consist of baby steps along with ideal multiplication. We look at these operations next.

3.2.1 Baby Steps and Ideal Reduction

Let $F = K(x, y)$ be a real quadratic function field corresponding to a hyperelliptic curve $C : y^2 + h(x)y = f(x)$. Let $\mathcal{O}$ be the integral closure of $K[x]$ in F. Consider a non-zero primitive $\mathcal{O}$-ideal $\mathfrak{a} = (a, b)$, for polynomials $a, b \in K[x]$. Let $\alpha = (b+y)/a \in F$ and note that α is irrational over $K(x)$. The completion $\widehat{F}_{\mathscr{P}_\infty}$ of F with respect to the place at infinity $\mathscr{P}_\infty$ is equal to $K((1/x))$, the field of power series in the variable $1/x$ [Sch96, §1]. Therefore, $F \subseteq K((1/x))$ and we can represent $\alpha \in F$ as a formal power series of Puiseux type, *i.e.* $\alpha = \sum_{i=-\infty}^{m} c_i x^i \in K((1/x))$ with $c_m \neq 0$ for some $m \in \mathbb{N}_0$. We define $\lfloor \alpha \rfloor = \sum_{i=0}^{m} c_i x^i$.

A **continued fraction** of the form

$$\alpha_0 + \cfrac{1}{\alpha_1 + \cfrac{1}{\alpha_2 + \cfrac{1}{\alpha_3 + \cdots}}}$$

will be represented in the compact notation of $[\alpha_0, \alpha_1, \alpha_2, \alpha_3, \ldots]$. We can apply Algorithm 3.9 to $\alpha \in K((1/x))$ to get a continued fraction expansion $\alpha = [\alpha_0, \alpha_1, \ldots]$ [BS96, pp. 75–79].

Algorithm 3.9 (Continued fraction algorithm). *The standard continued fraction algorithm in* $K((1/x))$.

Input: $\alpha \in K((1/x))$ and $n \in \mathbb{N}_0$.

Output: A continued fraction expansion $\alpha = [\alpha_0, \alpha_1, \ldots, \alpha_n]$.

1: $t_0 \leftarrow \alpha$

2: $\alpha_0 \leftarrow \lfloor t_0 \rfloor$

3: $i \leftarrow 0$

4: **while** $t_i \neq \alpha_i \ \wedge \ i < n$ **do**

5: $\quad i \leftarrow i + 1$

6: $\quad t_i \leftarrow 1/(t_{i-1} - \alpha_{i-1})$

7: $\quad \alpha_i \leftarrow \lfloor t_i \rfloor$

Expanding the operations in Algorithm 3.9 when $\alpha = (b + y)/a$ gives

$$\alpha_i = \left\lfloor \frac{b_i + \lfloor y \rfloor}{a_i} \right\rfloor \qquad (i \in \mathbb{N}_0)$$

with the polynomials $a_i, b_i \in K[x]$ computed using the following recursive formulae:

$$\begin{aligned} b_0 &= b & a_0 &= a \\ b_i &= \alpha_{i-1}a_{i-1} - b_{i-1} + h & a_i &= \frac{f - b_i^2 + b_i h}{a_{i-1}} \qquad (i \in \mathbb{N})\,. \end{aligned} \tag{3.4}$$

Performing the continued fraction algorithm in this way gives, for each step $i = 0, 1, 2, \ldots$, a pair of polynomials (a_i, b_i) that corresponds to the standard representation of a primitive $\mathcal{O}$-ideal $\mathfrak{a}_i = (a_i, b_i)$, where $\mathfrak{a}_0 = \mathfrak{a}$ [SW99, §2],[Zuc97a, §§1–2].

We now give some properties of continued fractions that can be found in Williams and Wunderlich [WW87, §2] or Hardy and Wright [HW79, pp. 130–141]. In the notation of Algorithm 3.9, we call $t_i = [\alpha_i, \alpha_{i+1}, \ldots]$ the **i-th complete quotient**. Using the polynomials from Equation (3.4), the i-th complete quotient is $t_i = (b_i + y)/a_i$ for any $i \in \mathbb{N}_0$. Then from Equation (3.4) and the curve equation $y^2 + hy = f$, we have

$$\frac{1}{t_i} = \frac{a_i}{b_i + y} = \frac{f - b_i^2 + b_i h}{a_{i-1}(b_i + y)} = \frac{(y - b_i + h)(y + b_i)}{a_{i-1}(b_i + y)} = \frac{y - b_i + h}{a_{i-1}} \qquad (i \in \mathbb{N})\,. \tag{3.5}$$

We define the **i-th convergent** of the continued fraction expansion as $[\alpha_0, \alpha_1, \ldots, \alpha_i] = p_i/q_i$, where p_i and q_i are given in the following recursive formulae:

$$\begin{aligned} p_{-2} &= 0\,, & p_{-1} &= 1\,, & p_i &= \alpha_i p_{i-1} + p_{i-2} \quad (i \in \mathbb{N}_0)\,, \\ q_{-2} &= 1\,, & q_{-1} &= 0\,, & q_i &= \alpha_i q_{i-1} + q_{i-2} \quad (i \in \mathbb{N}_0)\,. \end{aligned}$$

It follows that

$$\alpha = t_0 = \frac{\alpha_i p_{i-1} + p_{i-2}}{\alpha_i q_{i-1} + q_{i-2}} = \frac{t_i p_{i-1} + p_{i-2}}{t_i q_{i-1} + q_{i-2}} \qquad (i \in \mathbb{N}_0)\,. \tag{3.6}$$

The functions p_i and q_i also satisfy

$$p_{i-1}q_{i-2} - p_{i-2}q_{i-1} = (-1)^i \qquad (i \in \mathbb{N}_0)\,. \tag{3.7}$$

We will also define the following sequence:

$$\theta_1 = 1, \qquad \theta_{i+1} = \prod_{j=1}^{i} \frac{1}{t_j} \qquad (i \in \mathbb{N})\,. \tag{3.8}$$

Then from Equation (3.6) and induction on i we have the relation

$$\theta_{i+1} = (-1)^i(p_{i-1} - \alpha q_{i-1}) \qquad (i \in \mathbb{N}_0)\,. \tag{3.9}$$

Let $\overline{\theta} = \iota(\theta)$ denote the conjugate of $\theta \in F$ under the hyperelliptic involution ι. Then we have a **norm** defined as $\mathrm{N}(\theta) = \theta\overline{\theta}$ and from Equations (3.8), (3.4) and (3.5) we can derive

$$\mathrm{N}(\theta_{i+1}) = \theta_{i+1}\overline{\theta_{i+1}} = (-1)^i \frac{a_i}{a_0} \qquad (i \in \mathbb{N}_0)\,. \tag{3.10}$$

We will now show that the $\mathcal{O}$-ideals obtained in the continued fraction expansion of α are equivalent to each other. We follow a proof by Williams and Wunderlich given in the quadratic number field situation [WW87, §4]. Start by rewriting Equation (3.9)

in the following matrix format:

$$\begin{pmatrix}\theta_{i+1}\\ \theta_{i+2}\end{pmatrix} = X\begin{pmatrix}1\\ \alpha\end{pmatrix}, \qquad \text{where} \quad X = (-1)^{i+1}\begin{pmatrix}-p_{i-1} & q_{i-1}\\ p_i & -q_i\end{pmatrix}.$$

Since $\det(X) = \pm 1$ by Equation (3.7), we have $X \in \mathrm{GL}_2(K[x])$. Recall that we can represent an ideal by its non-unique $K[x]$-basis. Then applying Equation (3.1) we obtain the following equation of ideal bases:

$$(\theta_{i+1})\{1, \tfrac{1}{t_{i+1}}\} = \Big\{\theta_{i+1}, \tfrac{1}{t_{i+1}}\prod_{j=1}^{i}\tfrac{1}{t_i}\Big\} = \{\theta_{i+1}, \theta_{i+2}\} = \{1, \alpha\} = \{1, t_0\}\,.$$

We continue with this equation, simplifying as follows:

$$\begin{aligned}
(\theta_{i+1})\{1, \tfrac{1}{t_{i+1}}\} &= \{1, t_0\} \\
(\theta_{i+1})\big\{1, \tfrac{y-b_{i+1}+h}{a_i}\big\} &= \big\{1, \tfrac{b_0+y}{a_0}\big\} && \text{by Eqn. (3.5)} \\
(a_0\theta_{i+1})\{a_i, y - b_{i+1} + h\} &= (a_i)\{a_0, b_0 + y\} \\
(a_0\theta_{i+1})\{a_i, y + b_i - \alpha_i a_i\} &= (a_i)\{a_0, b_0 + y\} && \text{by Eqn. (3.4)} \\
(a_0\theta_{i+1})(a_i, b_i + y) &= (a_i)(a_0, b_0 + y) && \text{in standard representation} \\
(a_0\theta_{i+1})\mathfrak{a}_i &= (a_i)\mathfrak{a}_0 \\
(\theta_{i+1})\mathfrak{a}_i &= \big(\tfrac{a_i}{a_0}\big)\mathfrak{a}_0 \\
(\theta_{i+1})\mathfrak{a}_i &= \big((-1)^i\theta_{i+1}\overline{\theta_{i+1}}\big)\mathfrak{a}_0 && \text{by Eqn. (3.10)} \\
\mathfrak{a}_i &= \big((-1)^i\overline{\theta_{i+1}}\big)\mathfrak{a}_0\,. && (3.11)
\end{aligned}$$

Therefore, $\gamma = (-1)^i\overline{\theta_{i+1}}$ is a relative generator of $\mathfrak{a}_i$ with respect to $\mathfrak{a}_0 = \mathfrak{a}$ and we have that $\mathfrak{a}_i \sim \mathfrak{a}$ for every $i \in \mathbb{N}_0$.

Paulus and Stein showed that one could avoid the divisions in the formula for a_i from Equation (3.4) in odd characteristic fields [PS98, §4.2]. Algorithm 3.10 presents the optimized version, generalized to both even and odd characteristics, to compute equivalent ideals in the infrastructure of an ideal class [Ste01, §4.2],[Zuc97a, §2]. This algorithm is called the **baby step algorithm** and we will it denote by $\rho(\mathfrak{a}_i) = \mathfrak{a}_{i+1}$ for $i \in \mathbb{N}_0$.

Algorithm 3.10 (Infrastructure baby step ρ). *Baby step algorithm for computing in the infrastructure.*

Input: A primitive $\mathcal{O}$-ideal $\mathfrak{a}_i = (a_i, b_i)$ for some $i \in \mathbb{N}_0$, where $\mathcal{O} = K[C]$ for a real hyperelliptic curve $C : y^2 + hy = f$; if $i > 0$, then also the values d, a_{i-1}, r_{i-1}.

Output: A primitive ideal $\mathfrak{a}_{i+1} = (a_{i+1}, b_{i+1}) \sim \mathfrak{a}$.

1: **if** $i = 0$ **then**
2: $\quad d \leftarrow \lfloor y \rfloor$
3: $\quad r_0 \leftarrow b_0 + d \pmod{a}$
4: $\quad b_1 \leftarrow d - r_0 + h$
5: $\quad a_1 \leftarrow (f - b_1^2 + b_1 h)/a_0$
6: **else**
7: $\quad (\alpha_i, r_i) \leftarrow \text{divrem}(b_i + d, a_i)$ ▷ Division algorithm with remainder
8: $\quad b_{i+1} \leftarrow d - r_i + h$
9: $\quad a_{i+1} \leftarrow a_{i-1} + \alpha_i(r_i - r_{i-1})$

The following lemma shows that by applying the baby step algorithm on a primitive ideal, we will rapidly obtain an equivalent reduced ideal [JSS07b, §5].

Lemma 3.11 (Jacobson, Scheidler & Stein, 2007). *For any primitive $\mathcal{O}$-ideal $\mathfrak{a}_0 = (a_0, b_0)$, the ideal $\mathfrak{a}_{k+j} = \rho(\mathfrak{a}_{k+j-1})$ is reduced for all $j \geq 1$ and*

$$k = \max\left\{1,\ \left\lceil (\deg(a_0) - g)/2 \right\rceil\right\} .$$

Therefore, reducing a primitive ideal $\mathfrak{a}_0 = (a_0, b_0)$ is performed by repeating Algorithm 3.10 until $\deg a_i \leq g$. Moreover, the reduced ideals form a cycle. This is shown in Figure 3.12 and motivates our use of the notation ρ to denote this operation.

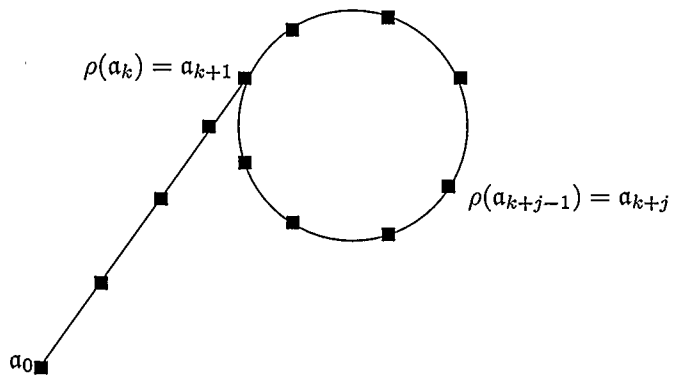

Figure 3.12. *Ideals resulting from the baby step algorithm from $\mathfrak{a}_0$ are represented by the dots which form the shape of ρ, where the equivalent reduced ideals are on the circle. The indices k and j are defined in Lemma 3.11. Note that if $\mathfrak{a}_0$ is reduced, then $k = 0$ and the picture would be a circle.*

Let $L(n) = \log n \log\log n$. Recall that to multiply or divide degree-n polynomials with remainder either can be performed in $O(nL(n))$ operations in K [AHU74, pp. 286–292]. Therefore, if $\deg \mathrm{N}(\mathfrak{a}) \in O(n)$, Algorithm 3.10 can perform a baby step on input $\mathfrak{a}$ in $O(nL(n))$ operations in K. Then according to Lemma 3.11, given a primitive $\mathcal{O}$ ideal $\mathfrak{a}$, we can obtain an equivalent reduced ideal in $O(n^2L(n))$ operations in K.

3.2.2 Distance and Closest Ideals

We can define an ordering of elements in the infrastructure using the notion of "distance" between ideals due to Shanks [Sha72b]. For reduced ideals $\mathfrak{a}$ and $\mathfrak{b}$ in the same ideal class of $\mathrm{Cl}(\mathcal{O})$, there is some $\ell \in \mathbb{N}_0$ such that $\mathfrak{a} = (a, b)$ and $\mathfrak{b} = \rho(\mathfrak{a}_{\ell-1}) = \mathfrak{a}_\ell = (a_\ell, b_\ell)$. Recall that $\mathfrak{a}_\ell = (\gamma)\mathfrak{a}$ for γ given in Equation (3.11). Then we define the **distance** between $\mathfrak{a}$ and $\mathfrak{b} = \mathfrak{a}_\ell$ as

$$\delta(\mathfrak{b}, \mathfrak{a}) = \delta(\mathfrak{a}_\ell, \mathfrak{a}) = \deg \gamma \,. \tag{3.12}$$

That is, the distance between equivalent ideals is the degree of a relative generator; the distance is only defined between equivalent ideals. When considering the distance from the trivial ideal $\mathcal{O}$, we will often simplify the notation of $\delta(\mathfrak{b}, \mathcal{O})$ by writing just $\delta(\mathfrak{b})$ [SW99, §3],[Zuc97a, §8].

In Algorithm 3.13 we update the baby step algorithm given in Algorithm 3.10 to also return the distance [Ste01, §§4.2–5]. When computing in the principal ideal class with $\mathfrak{e}_0 = \mathcal{O}$, the distance between each baby step is bounded by the following:

$$\begin{cases} 1 \leq \quad \delta(\mathfrak{e}_1) \quad \leq g+1 & \\ 1 \leq \quad \delta(\mathfrak{e}_i, \mathfrak{e}_{i-1}) \quad \leq g & \text{for } i \geq 2 \,. \end{cases} \tag{3.13}$$

The updated baby step algorithm has the same runtime complexity of $O(nL(n))$ operations in the field K.

Algorithm 3.13 (Infrastructure baby step ρ). *Baby step algorithm for computing in the infrastructure that also returns the relative distance.*

Input: A primitive $\mathcal{O}$-ideal $\mathfrak{a}_i = (a_i, b_i)$ for some $i \in \mathbb{N}_0$, where $\mathcal{O} = K[C]$ for a real

hyperelliptic curve $C : y^2 + hy = f$; if $i > 0$, then also the values d, a_{i-1}, r_{i-1}.

Output: A primitive ideal $\mathfrak{a}_{i+1} = (a_{i+1}, b_{i+1}) \sim \mathfrak{a}$ and the relative distance $\delta_{i+1} = \delta(\mathfrak{a}_{i+1}, \mathfrak{a}_i)$.

1: **if** $i = 0$ **then**
2: $d \leftarrow \lfloor y \rfloor$ ▷ The polynomial part of the root y
3: $r_0 \leftarrow b_0 + d \pmod{a}$ ▷ Compute the remainder of $(b_0 + d)/a$
4: $b_1 \leftarrow d - r_0 + h$
5: $a_1 \leftarrow (f - b_1^2 + b_1 h)/a_0$
6: $a_1 \leftarrow a_1 / \operatorname{sgn}(a_1)$ ▷ Make a_1 monic
7: $\delta_1 \leftarrow \max\{\deg(b_0 + d) - \deg a_0,\ 0\}$ ▷ Compute the distance $\delta(\mathfrak{a}_1, \mathfrak{a}_0)$
8: **else**
9: $(\alpha_i, r_i) \leftarrow \operatorname{divrem}(b_i + d, a_i)$ ▷ Division algorithm with remainder
10: $b_{i+1} \leftarrow d - r_i + h$
11: $a_{i+1} \leftarrow a_{i-1} + \alpha_i(r_i - r_{i-1})$
12: $a_{i+1} \leftarrow a_{i+1} / \operatorname{sgn}(a_{i+1})$ ▷ Make a_{i+1} monic
13: $\delta_{i+1} \leftarrow \deg \alpha_i$ ▷ Compute the distance $\delta(\mathfrak{a}_{i+1}, \mathfrak{a}_i)$

Using Algorithm 3.13 we can compute the closest ideal to a chosen small distance from another ideal using baby steps. Given a reduced $\mathcal{O}$-ideal $\mathfrak{a}$, the "closest" equivalent ideal to a distance d means that we find a reduced ideal $\mathfrak{b} \sim \mathfrak{a}$ with $\delta(\mathfrak{b}, \mathfrak{a}) \leq d$ such that there is no reduced ideal $\mathfrak{c} \sim \mathfrak{a}$ with $\delta(\mathfrak{b}, \mathfrak{a}) < \delta(\mathfrak{c}, \mathfrak{a}) \leq d$. We denote this property as $\delta(\mathfrak{b}, \mathfrak{a}) \lessapprox d$. Such an ideal can be found using Algorithm 3.14 and we denote the operation by ρ^*. If we wish to find the ideal closest to a distance d, Algorithm 3.14 runs in $O(dgL(g))$. In Section 3.3.4 we give a more efficient algorithm for computing closest ideals that is useful when the distance d is large.

Algorithm 3.14 (Infrastructure closest ideal ρ^*). *Compute the ideal that is closest to a given distance from an ideal.*

Input: A reduced $\mathcal{O}$-ideal $\mathfrak{a}$ and a desired distance $d \in \mathbb{N}$.

Output: A reduced ideal $\mathfrak{b} \sim \mathfrak{a}$ and the distance $\delta = \delta(\mathfrak{b}, \mathfrak{a})$ such that $\delta \lessapprox d$.

1: $i \leftarrow 0$, $\delta_0 \leftarrow 0$, $\mathfrak{a}_0 \leftarrow \mathfrak{a}$

2: **repeat**

3: $\quad i \leftarrow i + 1$

4: $\quad (\mathfrak{a}_i, \delta_i) \leftarrow \rho(\mathfrak{a}_{i-1}, \delta_{i-1})$ ▷ Compute a baby step with distance (Alg. 3.13)

5: **until** $\delta_i > d$

6: $\mathfrak{b} \leftarrow \mathfrak{a}_{i-1}$

7: $\delta \leftarrow \delta_{i-1}$

The distance can be used to find the regulator, as shown in the following theorem [Art21, p. 197]:

Theorem 3.15 (Artin, 1921). *There exists some $m \in \mathbb{N}$ such that $\mathfrak{a}_m = \rho(\mathfrak{a}_{m-1}) = \mathfrak{a}_0$. Then the regulator of $\mathcal{O}$ is $R_\mathcal{O} = \delta(\mathfrak{a}_m)$ for the smallest m satisfying $\rho(\mathfrak{a}_{m-1}) = \mathfrak{a}_0$ and $R_\mathcal{O} \neq 0$.*

Note that if $\rho(\mathfrak{a}_{m-1}) = \mathfrak{a}_0$, then it means the cycle of reduced ideals in Figure 3.12 is of length m. Figure 3.16 shows how the distance works in relation to the reduced ideals in an ideal class. Due to the cycle, we can reduce distances modulo the regulator $R_\mathcal{O}$. This is important because distances are unique and well-defined modulo $R_\mathcal{O}$.

While Theorem 3.15 gives a method to compute the regulator by performing many baby steps, there is a quicker method to find $R_\mathcal{O}$ by making a trade-off between baby steps and the "giant steps" that will be defined next.

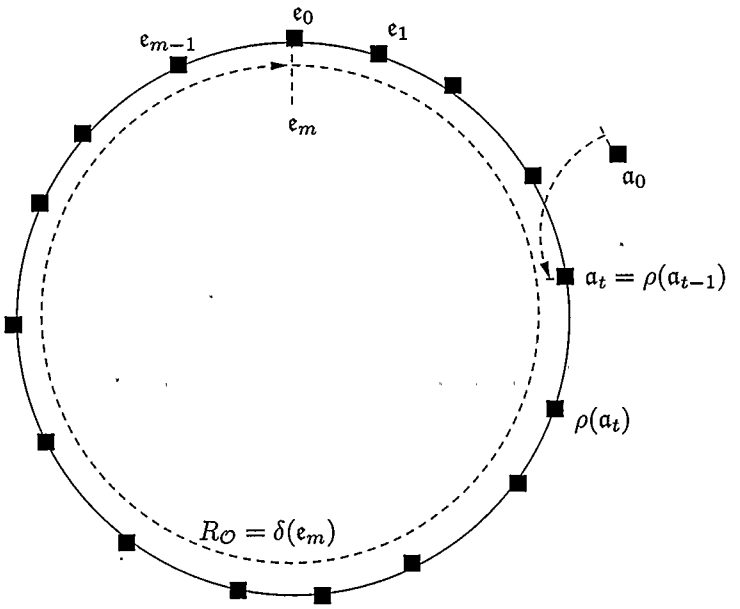

Figure 3.16. *We represent the cycle of reduced ideals in the principal ideal class by a circle. The squares on the circle denote the reduced ideals in the ideal class. One may perform baby steps from a principal non-reduced ideal $\mathfrak{a}_0$ to get to the closest reduced ideal $\mathfrak{a}_t$. Recall that once one gets to a reduced ideal, subsequent baby steps continue on the reduced ideals (*e.g. *$\rho(\mathfrak{a}_t)$). Rather than evenly spaced, the distance between each adjacent pair of reduced ideals varies slightly. The distance around the total circle is equal to the regulator $R_{\mathcal{O}}$.*

3.2.3 Giant Steps

Consider the infrastructure of the principal ideal class. Let $\mathfrak{a}$ and $\mathfrak{b}$ be two reduced principal $\mathcal{O}$-ideals. We can define an infrastructure operation $\mathfrak{a} * \mathfrak{b}$ that jumps over a large number of baby steps from $\mathfrak{b}$ by performing the following two operations:

i) **Composition:** Compute the product $\mathfrak{ab} = (s)\mathfrak{c}$ giving a primitive ideal $\mathfrak{c}$.

ii) **Reduction:** If necessary, use baby steps to find a reduced ideal $\mathfrak{c}_r$ equivalent to $\mathfrak{c}$, for some $r \in \mathbb{N}_0$.

Since $\mathfrak{a}$ and $\mathfrak{b}$ are principal, $\mathfrak{c}_r$ is also a reduced principal $\mathcal{O}$-ideal and, therefore, $\mathfrak{c}_r = \rho^u(\mathfrak{b})$ for some $u \in \mathbb{N}$. This operation in the infrastructure is performed via the **giant step algorithm** presented in Algorithm 3.17. We denote this operation by $*$.

Algorithm 3.17 (Infrastructure giant step $*$). *Giant step algorithm for computing in the infrastructure.*

Input: Two reduced principal $\mathcal{O}$-ideals $\mathfrak{a} = (a_\mathfrak{a}, b_\mathfrak{a})$ and $\mathfrak{b} = (a_\mathfrak{b}, b_\mathfrak{b})$ with associated distances $\delta(\mathfrak{a})$ and $\delta(\mathfrak{b})$.

Output: A reduced principal $\mathcal{O}$-ideal $\mathfrak{c} \sim \mathfrak{ab}$ and the distance $\delta_\mathfrak{c} = \delta(\mathfrak{c})$.

1: $(s)\mathfrak{c} = (s, a_\mathfrak{c}, b_\mathfrak{c}) \leftarrow \mathfrak{ab}$ ▷ Ideal multiplication, Alg. 3.4 or 3.5

2: $\delta_\mathfrak{c} \leftarrow \delta(\mathfrak{a}) + \delta(\mathfrak{b}) - \deg(s)$ ▷ Compute the new distance

3: $i \leftarrow 0$, $\mathfrak{c}_0 = (a_0, b_0) \leftarrow \mathfrak{c}$, $\delta_0 \leftarrow \delta_\mathfrak{c}$

4: **while** $\deg a_i > g$ **do** ▷ **Reduction**

5: $\quad i \leftarrow i + 1$

6: $\quad (\mathfrak{c}_i, \delta_i) = \big((a_i, b_i), \delta_i\big) \leftarrow \rho(\mathfrak{c}_{i-1}, \delta_{i-1})$ ▷ Baby steps with the distance

7: $\mathfrak{c} \leftarrow \mathfrak{c}_i$, $\delta_\mathfrak{c} \leftarrow \delta_i$

Let g be the genus of F and $L(n) = \log n \log\log n$. We can multiply two reduced ideals in $O(gL(g)\log g)$ operations in K according to Section 3.1.4. The result of the multiplication gives $\mathfrak{c}$ with $\deg a_\mathfrak{c} \in O(g)$. Reduction on $\mathfrak{c}$ will cost $O(g^2L(g))$ operations in K from Section 3.2.1. Then we can perform Algorithm 3.17 in $O(g^2L(g))$ operations in K.

Consider the distance resulting from the composition step. In Step 2 of Algorithm 3.17 we add the distance of $\mathfrak{a}$ to the distance of $\mathfrak{b}$ (we will consider the term $\deg(s)$ after). It is easy to see why if we consider $\mathfrak{a} = (\alpha)\mathcal{O}$ and $\mathfrak{b} = (\beta)\mathcal{O}$ with $\delta(\mathfrak{a}) = \deg\alpha$ and $\delta(\mathfrak{b}) = \deg\beta$. The result of the composition is $\mathfrak{ab} = (\alpha\beta)\mathcal{O}$ with $\delta(\mathfrak{ab}) = \deg(\alpha\beta) = \deg\alpha + \deg\beta$.

Now consider the distance after the reduction step. Note that we perform reduction on the primitive ideal $\mathfrak{c}$. That is, if the result of the multiplication $\mathfrak{ab}$ is not primitive, we must correct the distance for the rational coefficient $s \in \mathrm{Quot}(K[x])$. The resulting distance is given by

$$\delta(\mathfrak{a} * \mathfrak{b}) = \delta(\mathfrak{a}) + \delta(\mathfrak{b}) + \delta(\mathfrak{c}_r, \mathfrak{c}) - \deg(s)\ . \tag{3.14}$$

If we let $e = \delta(\mathfrak{c}_r, \mathfrak{c}) - \deg(s)$, then we always have $-2g \leq e \leq 0$. It is this "error" e that prevents the giant step from being associative and the infrastructure from being a group [SW99, §4B],[Zuc97a, §8]. Figure 3.18 shows the relationship between the distances involved in the giant step. But given that e is bounded to be insignificant in comparison to the distances of $\mathfrak{a}$ and $\mathfrak{b}$, we have $\delta(\mathfrak{a} * \mathfrak{b}) \approx \delta(\mathfrak{a}) + \delta(\mathfrak{b})$ and one can say that the giant step is "almost" associative. Therefore, we can use the infrastructure as if it is "almost" a group. In the next section we correct this operation to be associative.

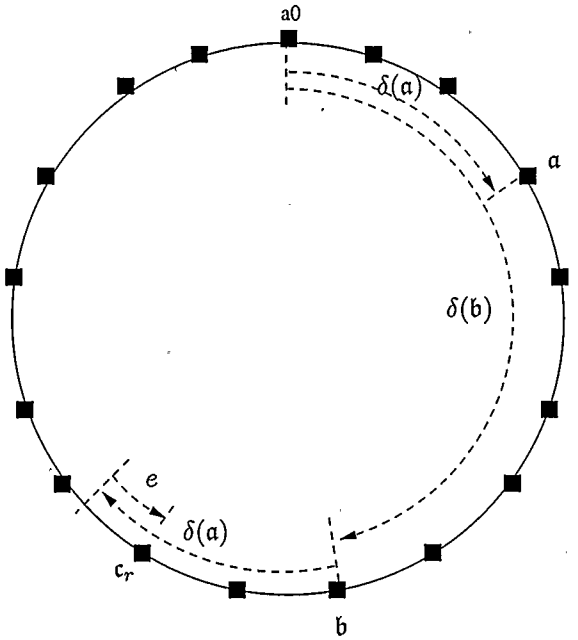

Figure 3.18. *For reduced principal ideals* $\mathfrak{a}$ *and* $\mathfrak{b}$*, the giant step* $\mathfrak{a} * \mathfrak{b} = \mathfrak{c}_r$*, where* $\mathfrak{c}_r$ *is at distance* $\delta(\mathfrak{c}_r) = \delta(\mathfrak{a}) + \delta(\mathfrak{b}) + e$.

3.2.4 Correcting the Giant Step

As mentioned in the last section, the giant step has a small error e such that for reduced principal $\mathcal{O}$-ideals $\mathfrak{a}$ and $\mathfrak{b}$, the result $\mathfrak{c} = \mathfrak{a} * \mathfrak{b}$ has distance $\delta(\mathfrak{a}) + \delta(\mathfrak{b}) + e$ from the trivial ideal $\mathcal{O}$. However, after performing the giant step, it is possible that e is large enough that there may exist another reduced principal ideal $\mathfrak{d}$ such that the distance $\delta(\mathfrak{d}) > \delta(\mathfrak{c})$ yet $\delta(\mathfrak{d}) \leq \delta(\mathfrak{a}) + \delta(\mathfrak{b})$. That is, $\mathfrak{d}$ is closer to the distance $\delta(\mathfrak{a}) + \delta(\mathfrak{b})$ (see Figure 3.19). Therefore, we may have to correct the giant step result $\mathfrak{c} = \mathfrak{a} * \mathfrak{b}$ to $\mathfrak{d}$ using baby steps.

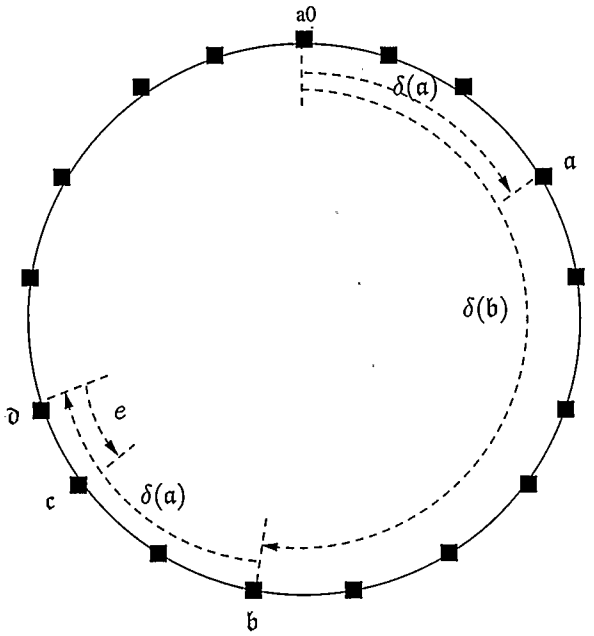

Figure 3.19. *For principal ideals* $\mathfrak{a}$ *and* $\mathfrak{b}$*, the giant step* $\mathfrak{a} * \mathfrak{b} = \mathfrak{c}$ *needs to be corrected to* $\mathfrak{d}$ *since the error term* $|e|$ *is too large thus making* $\mathfrak{c}$ *not as close to the distance* $\delta(\mathfrak{a}) + \delta(\mathfrak{b})$ *as* $\mathfrak{d}$*.*

An algorithm to perform the giant step together with the correction step in the infrastructure of a real quadratic function field was given by Scheidler, Stein and Williams [SSW96]. This **corrected giant step** is presented in Algorithm 3.20. We denote the corrected giant step using the operator $\circledast$.

Algorithm 3.20 (Infrastructure corrected giant step $\circledast$). *Corrected giant step algorithm for computing in the infrastructure.*

Input: Two reduced principal $\mathcal{O}$-ideals $\mathfrak{a}$ and $\mathfrak{b}$ with $\delta(\mathfrak{a})$ and $\delta(\mathfrak{b})$.

Output: A reduced principal $\mathcal{O}$-ideal $\mathfrak{c} \sim \mathfrak{a}\mathfrak{b}$ and $\delta(\mathfrak{c})$ such that $\mathfrak{c}$ has the greatest distance satisfying $\delta(\mathfrak{c}) \leq \delta(\mathfrak{a}) + \delta(\mathfrak{b})$.

1: $(\mathfrak{c}', \delta_{\mathfrak{c}}') \leftarrow \mathfrak{a} * \mathfrak{b}$ ▷ Compute the giant step and distance (Alg. 3.17)

2: $e \leftarrow \delta'_{\mathfrak{c}} - \delta(\mathfrak{a}) - \delta(\mathfrak{b})$ ▷ Compute the error term (Eqn. 3.14)

3: $(\mathfrak{c}, \delta_{\mathfrak{c}}) \leftarrow \rho^*(\mathfrak{c}', -e)$ ▷ Closest ideal to the distance $-e$ from $\mathfrak{c}'$ (Alg. 3.14)

4: $\delta(\mathfrak{c}) \leftarrow \delta'_{\mathfrak{c}} + \delta_{\mathfrak{c}}$

Recall that Section 3.2.3 bounded e as $-2g \leq e \leq 0$. Then Algorithm 3.20 runs in $O(g^2 L(g))$ operations in K, dominated by the cost of the giant step.

3.2.5 The Baby-Step Giant-Step Algorithm

Recall from Schmidt's class number relation (Lemma 3.7) that for a real quadratic function field F, the class number h_F of the degree-zero divisor class group $\mathrm{Cl}^0(F)$ and the class number $h_{\mathcal{O}}$ of the ideal class group $\mathrm{Cl}(\mathcal{O})$ are related via the regulator $R_{\mathcal{O}}$ as $h_F = h_{\mathcal{O}} R_{\mathcal{O}}$. As mentioned earlier, the regulator $R_{\mathcal{O}}$ can be computed by performing a combination of baby steps and giant steps. This **baby-step giant-step algorithm** computes and stores the result of a number of baby steps, then performs giant steps until a match is found. The baby-step giant-step algorithm is a time/memory trade-off originally proposed by Shanks [Sha71] in the context of real quadratic number fields. It was specified for real quadratic function fields over a constant field of odd characteristic by Stein and Williams [SW99, §3C] and in the even characteristic case by Zuccherato [Zuc97a, §9]. We present their method generalized for both characteristics in Algorithm 3.21.

Algorithm 3.21 (Regulator baby-step giant-step). *Baby-step giant-step algorithm for computing the regulator.*

Input: The genus g of a real quadratic function field F over a field $K = \mathbb{F}_q$, and the quadratic order $\mathcal{O}$ of F.

Output: The regulator $R_{\mathcal{O}}$.

1: $t \leftarrow \lfloor \frac{3}{2} q^{\frac{g+1}{2}} \rfloor$

2: $\mathfrak{e}_0 \leftarrow (1,0) = \mathcal{O}$ and store $(\mathfrak{e}_0, 0)$ ▷ Store the polynomial pair representing $\mathfrak{e}_0$

3: $\delta_0 \leftarrow 0$

4: **for** i **from** 1 **to** t **do**

5: $\quad (\mathfrak{e}_i, \delta_i) = \big((a_i, b_i), \delta_i\big) \leftarrow \rho(\mathfrak{e}_{i-1}, \delta_{i-1})$ ▷ Perform a baby step (Alg. 3.13)

6: $\quad$ **if** $a_i \in K^*$ **then** ▷ Check if a_i is a trivial unit of $\mathcal{O}$

7: $\quad\quad$ **return** $R_{\mathcal{O}} \leftarrow \delta_i$

8: $\quad$ Store $(\mathfrak{e}_i, \delta_i)$ ▷ Store the polynomial pair representing $\mathfrak{e}_i$

9: $\mathfrak{b}_0 \leftarrow \mathfrak{e}_t$, $\delta'_0 \leftarrow \delta_t$, $j \leftarrow 0$

10: **repeat**

11: $\quad j \leftarrow j + 1$

12: $\quad (\mathfrak{b}_j, \delta'_j) \leftarrow (\mathfrak{e}_t, \delta_t) * (\mathfrak{b}_{j-1}, \delta'_{j-1})$ ▷ Perform a giant step (Alg. 3.17)

13: **until** $\mathfrak{b}_j = \mathfrak{e}_k$ for some $k \in \{0, \ldots, t\}$ ▷ Check if $\mathfrak{b}_j$ is a stored value

14: **return** $R_{\mathcal{O}} \leftarrow \delta'_j - \delta_k$

The ideal/distance pairs should be stored in a hash table to allow for efficient look-up. The asymptotic runtime complexity of BSGS given in Algorithm 3.21 is bounded by $O(q^{\frac{g+1}{2}} g^2 L(g))$ operations in K due to the value of t, which is exponential in $g \log q$. The runtime of the algorithm can be improved in practice by using properties of symmetry and conjugate ideals, details that can be found Zuccherato [Zuc97a, §9]. Further optimizations were provided by Teske and Stein that take advantage of the fact that baby steps are significantly faster than giant steps in the infrastructure [TS05, §2.5]. However, even using these improvements, the runtime of the baby step giant step algorithm is still exponential in $g \log q$.

Another limiting factor of Algorithm 3.21 is that it requires $O(q^{\frac{g+1}{2}})$ space. Teske and Stein discuss how baby step giant step can be implemented in a space-restricted environment [TS05, §§2.4, 2.5.2, 3.2]. A couple of generic algorithms given by Pollard can be adapted to regulator computation and have the same asymptotic runtime complexity as baby step giant step, but use significantly less memory [Pol78]. One of Pollard's methods has been implemented in real quadratic function fields by Stein and Teske [ST02b]

Stein and Williams derived a more efficient method for computing the regulator by performing two steps of BSGS. First, they use BSGS to search for $R_{\mathcal{O}} < G$ for an upper bound G. If $R_{\mathcal{O}}$ is not found, then the second step estimates the product of the class number and regulator via analytic methods[6] giving an interval in which to perform a BSGS search for $h_{\mathcal{O}}R_{\mathcal{O}}$ [SW99, §4]. In a subsequent paper, Stein and Williams pointed out that if one obtains $h_{\mathcal{O}}R_{\mathcal{O}}$ from the second step of their method, the regulator can be found by factoring the product and using the infrastructure to determine the smallest divisor that gives $R_{\mathcal{O}}$.[7] The total complexity of their algorithm for computing the regulator was $O(q^{\lambda}g^2L(g))$ operations in K, where $\lambda = (2g-1)/5$ if $g \equiv 3 \pmod 5$ or $\lambda = 2g/5$ otherwise [SW98, §§5.1]. Improved estimates for $h_{\mathcal{O}}R_{\mathcal{O}}$ to reduce the interval for searching in the second step were provided by Stein and Teske [ST02a, §5].

In Chapter 4 we will show how the index calculus algorithm allows one to compute the regulator in time and space subexponential in $\log q$ when the genus g is large.

[6] Similar analytic methods for estimating $h_{\mathcal{O}}R_{\mathcal{O}}$ are given in Section 4.3.7.

[7] We describe this factoring technique for finding the regulator from a multiple of $R_{\mathcal{O}}$ in Section 4.3.6.

3.3 Efficient Arithmetic in the Infrastructure

The algorithms presented in the previous section for the baby step and giant step in the infrastructure of a real quadratic function field use an approach based on continued fraction expansions. Recent work has produced techniques for improving the efficiency of both algorithms.

In cryptographic protocols, it is often required to perform an exponentiation operation consisting of a scalar multiple of giant steps. Many techniques such as the "square-and-multiply" algorithm can reduce the total number of giant steps performed, but recent proposals show that these techniques can be combined with a reordering of operations to improve the efficiency further. This section examines both improvements to the giant step operation and efficient exponentiation.

3.3.1 NUCOMP

The NUCOMP algorithm was originally proposed by Shanks [Sha89] for computations involving binary quadratic forms, or equivalently, ideals in imaginary quadratic number fields. NUCOMP was adapted to real quadratic number fields by van der Poorten [vdP03]. Jacobson and van der Poorten [JvdP02] discovered that NUCOMP could be applied to ideal computations in function fields. Further improvements to reduce the operands for quadratic forms were proposed by Atkin and adapted to function fields by Jacobson, Scheidler and Stein [JSS07b].

The standard algorithm for giant steps in the infrastructure perform composition and reduction sequentially. The idea of NUCOMP is to perform reductions before the composition, thereby allowing one to work with polynomials of smaller degree. The complete algorithm as well as detailed explanations can be found in the work of

Jacobson, Scheidler and Stein [JSS07b, §§8–9]. The algorithms given can be used in replacement of Algorithm 3.17 to improve computational efficiency.

3.3.2 Explicit Formulae

Whereas the previous algorithms work in general for all hyperelliptic curves, explicit formulae focus on hyperelliptic curves of a fixed, small genus to tweak the operations for a reduction in complexity. The basic idea is to express operations in terms of forumlae for polynomial coefficients, instead of in terms of generic polynomial arithmetic. Implementations are designed to use the efficient explicit formulae for special cases and otherwise use a general algorithm, with the hope that the special cases occur often enough to provide a significant performance improvement overall.

Complexity results for explicit formulae differentiate between types of field operations. Addition in the field K is considered to be negligible and not included in most analyses. Multiplications are often separated from squarings as well as field inversions, the latter being the most costly.

Let the hyperelliptic curve C over a field K be given by the equation $C : y^2 + h(x)y = f(x)$, where f_i represents the i-th coefficient of the polynomial $f(x)$ (*e.g.* the constant term is f_0). The first explicit formulae for real quadratic function fields were provided very recently by Erickson *et al.* [EJS+07]. They give formulae for genus $g = 2$ under the assumption that $\operatorname{char} K > 3$. The latter assumption allows a transformation $x \mapsto x - f_5/6$ that eliminates the x^5 term in $f(x)$.

3.3.3 Fast Giant Step Exponentiation

Many techniques have been proposed for improving the efficiency when computing a scalar multiple of an operation. They involve representing m in some sort of digit encoding and then performing a square-and-multiply algorithm. Most of these techniques can be applied directly to compute $m \in \mathbb{N}$ corrected giant steps. If we interpret corrected giant steps as multiplications, then our goal is to compute the **exponentiation** of an ideal $\mathfrak{a}$. We denote this as

$$\mathfrak{a} ** m = \underbrace{\mathfrak{a} \circledast \mathfrak{a} \circledast \cdots \circledast \mathfrak{a}}_{m} .$$

The simplest (non-naïve) technique for exponentiation uses the **binary representation** of m, written as

$$m = \sum_{i=0}^{\ell} b_i 2^i , \qquad \text{with } b_i \in \{0,1\} \text{ for } 0 \leq i \leq \ell - 1 \text{ and } b_\ell = 1 .$$

Note that the length of the binary representation is $\ell = \lfloor \log m \rfloor$. Using binary representation is easy since computers represent integers in binary by default. The **square-and-multiply algorithm** is the standard method for computing exponentiations and we present it for infrastructure exponention in Algorithm 3.22.

Algorithm 3.22 (Infrastructure exponentiation $**$). *Square-and-multiply exponentiation algorithm for computing the result of m (corrected) giant steps in the infrastructure with m given in binary form.*

Input: A reduced principal $\mathcal{O}$-ideal $\mathfrak{a}$ with its distance $\delta_\mathfrak{a} = \delta(\mathfrak{a})$ and an exponent $m = \sum_{i=0}^{\ell} b_i 2^i \in \mathbb{N}$ given in binary.

Output: A reduced principal $\mathcal{O}$-ideal $\mathfrak{b} \sim \mathfrak{a}$ of distance $\delta_\mathfrak{b} = \delta(\mathfrak{b}) \leq m\delta(\mathfrak{a})$ such that $\delta(\mathfrak{b})$ is maximal.

1: $(\mathfrak{b}, \delta_\mathfrak{b}) \leftarrow (\mathfrak{a}, \delta_\mathfrak{a})$
2: **for** i **from** $\ell - 1$ **to** 0 **do**
3: $\quad (\mathfrak{b}, \delta_\mathfrak{b}) \leftarrow (\mathfrak{b}, \delta_\mathfrak{b}) \circledast (\mathfrak{b}, \delta_\mathfrak{b})$ ▷ "Square" $\mathfrak{b}$ (Alg. 3.20)
4: $\quad$ **if** $b_i = 1$ **then**
5: $\quad\quad (\mathfrak{b}, \delta_\mathfrak{b}) \leftarrow (\mathfrak{b}, \delta_\mathfrak{b}) \circledast (\mathfrak{a}, \delta_\mathfrak{a})$ ▷ "Multiply" $\mathfrak{b}$ by $\mathfrak{a}$ (Alg. 3.20)

We define the **Hamming weight** of m as the number of non-zero entries in a particular digit representation of m. It should be easy to see that the square/multiply algorithm performs fewer "multiplications" when the Hamming weight of m is small.

If we have knowledge of the regulator $R_\mathcal{O}$, then we can take advantage of the fact that we can compute the distance of an ideal in the infrastructure in the opposite direction. Because of the symmetry of the infrastructure, the conjugate ideal $\overline{\mathfrak{a}}$ has distance $\delta(\overline{\mathfrak{a}}) = R_\mathcal{O} - \delta(\mathfrak{a})$. Since computing $\overline{\mathfrak{a}}$ is free, we can use a represention of m called **signed-binary representation**. This corresponds to

$$m = \sum_{i=0}^{\ell+1} c_i 2^i, \qquad \text{with } c_i \in \{-1, 0, 1\} \text{ for } 0 \leq i \leq \ell \text{ and } c_{\ell+1} = 1\,.$$

In particular, m in signed-binary representation is in **non-adjacent form (NAF)** if no consecutive pair (c_i, c_{i+1}) are both non-zero. This representation in NAF is unique for any $m \in \mathbb{N}$.[8] Written in NAF, m has at most $\ell + 1$ digits and, due to the restriction that no pair of adjacent digits are both non-zero, the Hamming weight of m is expected to be $\frac{1}{3}(\ell+1)$ in NAF, versus $\frac{1}{2}\ell$ for binary representation. In fact, the

[8]For proofs of the existence and uniqueness of NAF, see Reitwiesner [Rei60, pp. 246–248]

Hamming weight of an integer in NAF is guaranteed to be minimal among all possible signed-binary representations. One can recode m from binary to NAF representation using Algorithm 3.23 [Doc06, pp. 150–152]. Then the square-and-multiply algorithm is slightly modified as shown in Algorithm 3.24 to work with the scalar exponent in NAF.

Algorithm 3.23 (Binary to NAF recoding). *Compute the non-adjacent form (NAF) representation of* $m \in \mathbb{N}$.

Input: The binary representation $m = \sum_{i=0}^{\ell} b_i 2^i$ with $b_\ell \neq 0$.

Output: The NAF representation $m = \sum_{i=0}^{\ell+1} c_i 2^i$.

1: $r_0 \leftarrow 0$, $b_{\ell+1} \leftarrow 0$, $b_{\ell+2} \leftarrow 0$

2: **for** i **from** 0 **to** $\ell + 1$ **do**

3: $\quad r_{i+1} \leftarrow \lfloor (r_i + b_i + b_{i+1})/2 \rfloor$

4: $\quad c_i \leftarrow r_i + b_i - 2r_{i+1}$

Algorithm 3.24 (NAF-based infrastructure exponentiation $**$). *Square-and-multiply exponentiation algorithm for computing the result of* m *(corrected) giant steps in the infrastructure with* m *given in non-adjacent form (NAF).*

Input: A reduced principal $\mathcal{O}$-ideal $\mathfrak{a}$ with its distance $\delta_\mathfrak{a} = \delta(\mathfrak{a})$ and an exponent $m = \sum_{i=0}^{\ell+1} c_i 2^i \in \mathbb{N}$ given in NAF. We also assume we have the regulator $R_\mathcal{O}$

Output: A reduced principal $\mathcal{O}$-ideal $\mathfrak{b} \sim \mathfrak{a}$ of distance $\delta_\mathfrak{b} = \delta(\mathfrak{b}) \leq m\delta(\mathfrak{a})$ such that $\delta(\mathfrak{b})$ is maximal.

1: $(\mathfrak{b}, \delta_\mathfrak{b}) \leftarrow (\mathfrak{a}, \delta_\mathfrak{a})$

2: **for** i **from** $\ell + 1$ **to** 0 **do**

3: $\quad (\mathfrak{b}, \delta_\mathfrak{b}) \leftarrow (\mathfrak{b}, \delta_\mathfrak{b}) \circledast (\mathfrak{b}, \delta_\mathfrak{b})$ ▷ "Square" $\mathfrak{b}$ using the corrected giant step

4: $\quad$ **if** $c_i = 1$ **then**

5: $(\mathfrak{b}, \delta_{\mathfrak{b}}) \leftarrow (\mathfrak{b}, \delta_{\mathfrak{b}}) \circledast (\mathfrak{a}, \delta_{\mathfrak{a}})$ ▷ Perform a corrected giant step with $\mathfrak{a}$

6: **else if** $(c_i = -1)$ **then**

7: $(\mathfrak{b}, \delta_{\mathfrak{b}}) \leftarrow (\mathfrak{b}, \delta_{\mathfrak{b}}) \circledast (\overline{\mathfrak{a}}, R_{\mathcal{O}} - \delta_{\mathfrak{a}})$ ▷ Corrected giant step with conjugate of $\mathfrak{a}$

In practice, exponentiation in the infrastructure can be improved even more. Note that each corrected giant step operation may perform a number of baby steps to ensure the intermediate result is valid. Jacobson, Scheidler and Stein [JSS07a] gave heuristics that make the distance between consecutive baby steps or giant steps precise.

Heuristic 3.25 (Jacobson, Scheidler & Stein, 2005–2007). *Let C be a real hyperelliptic curve of genus g over the finite field $K = \mathbb{F}_q$. Let $F = K(C)$ be the function field of C, and let $\mathcal{O} = K[C]$ be its coordinate ring. Then for two reduced principal $\mathcal{O}$-ideals $\mathfrak{a}$ and $\mathfrak{b}$ and sufficiently large q, the following properties hold with probability $1 - O(q^{-1})$:*

i) If $\mathfrak{c} = \rho(\mathfrak{a}, 0)$ is the result of a baby step from $\mathfrak{a}$, then the distance between $\mathfrak{a}$ and $\mathfrak{c}$ is always equal to $\delta(\mathfrak{c}) - \delta(\mathfrak{a}) = 1$;

ii) If $\mathfrak{c} = \mathfrak{a} * \mathfrak{b}$ is the result of a giant step, then the resulting distance $\delta(\mathfrak{c})$ has an error value always equal to $e = -\lceil g/2 \rceil$.

These heuristics allow one to avoid using the corrected version of the giant step by performing a few baby steps at the beginning of the exponentiation. Therefore, one avoids the adjustment baby steps in each intermediate giant step. We refer the reader to Jacobson, Scheidler and Stein [JSS07a, §3.3] for the presentation of these heuristic methods.

3.3.4 Fast Closest Ideals

Often we want to compute the closest ideal to a chosen distance from another ideal. Recall from Section 3.2.2 that given a reduced $\mathcal{O}$-ideal $\mathfrak{a}$, the "closest" equivalent ideal to a distance d means that we find a reduced ideal $\mathfrak{b} \sim \mathfrak{a}$ with $\delta(\mathfrak{b}, \mathfrak{a}) \leq d$ such that there is no reduced ideal $\mathfrak{c} \sim \mathfrak{a}$ with $\delta(\mathfrak{b}, \mathfrak{a}) < \delta(\mathfrak{c}, \mathfrak{a}) \leq d$. We denote this property as $\delta(\mathfrak{b}, \mathfrak{a}) \lessapprox d$.

Algorithm 3.14 computed the closest ideal by performing baby steps from $\mathfrak{a}$, keeping track of the distance travelled, and stopping when we reached the desired distance d. We can improve upon this naïve method by using giant step exponentiation. Algorithm 3.26 presents this improved method [JSS07a, §3.2].

Algorithm 3.26 (Infrastructure closest ideal ρ^*). *Computing the closest equivalent ideal to a given distance from a given ideal.*

Input: A reduced principal $\mathcal{O}$-ideal $\mathfrak{a}$ with its distance $\delta_{\mathfrak{a}} = \delta(\mathfrak{a})$ and a distance d.

Output: A reduced principal $\mathcal{O}$-ideal $\mathfrak{b} \sim \mathfrak{a}$ and the distance $\delta = \delta(\mathfrak{b}, \mathfrak{a})$ such that $\delta \lessapprox d$.

1: $m \leftarrow \lfloor d/\delta_{\mathfrak{a}} \rfloor$
2: $(\mathfrak{b}, \delta) \leftarrow (\mathfrak{a}, 0) ** m$
3: **while** $\delta \leq d$ **do**
4: $\quad (\mathfrak{b}', \delta') \leftarrow \rho(\mathfrak{b}, \delta)$
5: $\quad$ **if** $\delta \leq d$ **then**
6: $\quad\quad \mathfrak{b} \leftarrow \mathfrak{b}'$, $\delta \leftarrow \delta'$

Similar heuristic techniques to the fast exponentiation algorithm can be applied to finding closest ideals. See Jacobson, Scheidler and Stein [JSS07a, §3.3] for a discussion

of these heuristic methods.

Chapter 4

Index Calculus in Real Quadratic Function Fields

Cryptographic protocols proposed by Scheidler, Stein and Williams [SSW96] and Jacobson, Scheidler and Stein [JSS07a] are based on a computational problem in a real quadratic function field called the infrastructure discrete logarithm problem. Discrete logarithms arise in other areas, such as the multiplicative group of integers modulo a prime, where the fastest known algorithm to solve an instance is known as the index calculus algorithm. There are other computational problems of interest in real quadratic function fields, such as finding the regulator and computing the ideal class number. When the genus is large, we will show that index calculus is the fastest known method for solving these problems, too. In this chapter we discuss how the index calculus algorithm can be formulated in the infrastructure and present a heuristic analysis of its complexity.

We begin this chapter by formally stating a number of computational problems in the infrastructure in Section 4.1. Section 4.2 introduces the framework of the

index calculus algorithm. Details of how the index calculus algorithm is applied in the infrastructure are presented in Section 4.3, including explicit algorithms and a complexity analysis for each of the main computational problems.

4.1 Computational Problems in the Infrastructure

Like any interesting mathematical structure, there are a number of computationally difficult problems that are associated with the infrastructure of a real quadratic function field. The complexity of an algorithm is said to be **expected polynomial time** if its running time is of the form $O(\log^c n)$ on average, where $\log n$ is the number of bits required to express the input n and $c \in \mathbb{R}_{\geq 0}$ is a constant.[1] We define a **computationally difficult problem** as a problem for which no expected polynomial-time algorithm is known to solve a random instance.

For the problems we are considering there is no proof to show that they cannot be solved in polynomial time. The basis for the claim is that after many years of study, mathematicians and computer scientists have been unable to find an algorithm that provides a solution in polynomial time. This section provides an overview of the most common problems of interest in the infrastructure of a real quadratic function field.

4.1.1 Properties of the Ideal Class Group

Let F be a real quadratic function field defined by $y^2 + hy - f$ for polynomials $h, f \in K[x]$. Let $\mathcal{O}$ be the quadratic order of F. Recall from Section 3.1.5, the ideal class group $\mathrm{Cl}(\mathcal{O})$ is a set of equivalence classes of ideals. The number of ideal classes,

[1] If $c = 0$, then the algorithm is said to be **constant** with respect to the input size.

i.e. the order of the ideal class group, is the ideal class number $h_{\mathcal{O}}$. Computing the ideal class number is believed to be a computationally difficult problem.

Problem 4.1 (Class number computation). *Given polynomials f and h, compute the class number of the ideal class group* $\mathrm{Cl}(\mathcal{O})$.

In real quadratic function fields, the ideal class number $h_{\mathcal{O}}$ does not immediately give us the divisor class number h_F (*cf.* Theorem 3.7). One also requires the regulator $R_{\mathcal{O}}$, the degree of the fundamental unit $\eta \in \mathcal{O}^*$. As shown in Section 3.2.2, the regulator also gives the length of the cycle of reduced ideals in the infrastructure of an ideal class. Since we can reduce distances modulo $R_{\mathcal{O}}$, it is important for efficient computations in the infrastructure. However, computing the regulator is also believed to be computationally difficult.

Problem 4.2 (Regulator computation). *Given polynomials f and h, compute the regulator $R_{\mathcal{O}}$ of $\mathcal{O}$.*

One of the central theorems in group theory is the following:

Theorem 4.3 (Fundamental theorem of Abelian groups). *Every finitely generated Abelian group is isomorphic to a unique decomposition of the following form:*

$$\mathbb{Z}^n \oplus \mathbb{Z}/s_1\mathbb{Z} \oplus \mathbb{Z}/s_2\mathbb{Z} \oplus \cdots \oplus \mathbb{Z}/s_t\mathbb{Z}\,,$$

where $n \geq 0$ and $s_i | s_{i+1}$ for $1 \leq i < t$.

The above decomposition is called the **structure** of the group. Since the ideal class group $\mathrm{Cl}(\mathcal{O})$ is a finite Abelian group, it has a structure as given in Theorem 4.3 with $n = 0$. It is of interest, and believed to be computationally difficult, to determine this group structure.

Problem 4.4 (Determine class group structure). *Given polynomials f and h, compute the group structure of the ideal class group* $\mathrm{Cl}(\mathcal{O})$.

4.1.2 Discrete Logarithm Problem

As mentioned in Chapter 1, the discrete logarithm problem is a computationally difficult problem that is used as the basis for many public key cryptosystems. In a generic multiplicative group G of large prime order,[2] the **discrete logarithm problem** (DLP) is defined as follows:

Problem 4.5 (Generic group DLP). *Let g be an element of order n in G. Then given an element $h = g^b$ for some unknown positive integer $b < n$, find b.*

Often, g is chosen to be a generator of the group G, so the order of g is equal to the size of G. The choice of G affects the difficulty of solving Problem 4.5. For cryptographic applications G must be chosen to be sufficiently large so that g can have a large order, but how large is determined by the fastest known algorithm to solve Problem 4.5 in the type of group.

Even though the infrastructure of a real quadratic function field does not form a group, we can still define a similar discrete logarithm problem in infrastructure. The definition and use of the infrastructure DLP for cryptography was proposed by Scheidler, Stein and Williams [SSW96] based on a similar proposal for the infrastructure of a real quadratic number field by Buchmann and Williams [BW90].

Problem 4.6 (Infrastructure DLP). *Given a reduced principal $\mathcal{O}$-ideal $\mathfrak{b}$, find the distance $\delta(\mathfrak{b})$.*

[2]Using a multiplicative group G is not a restriction; we only specify that G is multiplicative rather than additive in order to have consistent notation. An example of an additive group that admits an interesting DLP is the group of points on an elliptic curve.

Similar to Problem 4.5, the difficulty of the infrastructure DLP is directly affected by the size of the regulator $R_{\mathcal{O}}$.

In many cryptographic protocols, $\mathfrak{b}$ is computed as $\mathfrak{b} = \mathfrak{c} * b$ for some publically agreed-upon principal $\mathcal{O}$-ideal $\mathfrak{c} \neq \mathcal{O}$ and randomly chosen private value $b \in \mathbb{N}$ satisfying $1 \leq b \leq R_{\mathcal{O}}$. In such a situation, one may want to find the integer b. From the definition of the giant step exponentiation, $\mathfrak{b}$ has distance closest to b. So given a solution $\delta = \delta(\mathfrak{b})$ to an instance of the infrastructure DLP, we can compute $\rho^*(\mathcal{O}, b')$ for $b' = \delta, \delta - 1, \ldots$ until we get $\mathfrak{b}$. Note that according to Equation (3.13), there are at most g choices for b'.

4.1.3 Subexponential Algorithms

An algorithm is called **exponential** if it is not polynomial time, but has a running time in $O(c^{f(n)})$ for some constant $c \in \mathbb{R}_{>1}$ and some function f that is polynomial in $\log n$. To better classify algorithms, a distinction is made between a **fully-exponential** algorithm and a **subexponential** algorithm whose running time is of the form $O(e^{o(\log n)})$, where e is base of the natural logarithm. To express the running time of an algorithm that is subexponential in $\log n$, we use the following convenient notation:

$$\mathbf{L}_n(\alpha, c) = e^{c(\log n)^{\alpha}(\log\log n)^{1-\alpha}},$$

where $c, \alpha \in \mathbb{R}$ are constants satisfying $c > 0$ and $0 < \alpha < 1$. For an algorithm with running time $O\big(\mathbf{L}_n(\alpha, c)\big)$, if $\alpha = 0$, then the algorithm is polynomial in $\log n$; also, if $\alpha = 1$, then the algorithm is fully-exponential in $\log n$. If $\alpha = 1/2$, as will be common

throughout this chapter, we simplify this notation as

$$\mathbf{L}_n(c) = e^{c\sqrt{\log n \log\log n}}.$$

Note that we have the following simple relations that we will use to simplify expressions involving subexponential terms:

$$O\big(\mathbf{L}_n(c_1)\big) \cdot O\big(\mathbf{L}_n(c_2)\big) \subseteq O\big(\mathbf{L}_n(c_1 + c_2)\big) \qquad \text{and}$$
$$O\big(\mathbf{L}_n(c_1)\big) + O\big(\mathbf{L}_n(c_2)\big) \subseteq O\big(\mathbf{L}_n(\max\{c_1, c_2\})\big).$$

All of the problems we have described in this section can be solved in expected subexponential time when the genus of the function field is large. This algorithm will be described next.

4.2 Introduction to Index Calculus

The technique known as "index calculus" has been applied to many computationally difficult problems. In many situations index calculus turns out to have an expected subexponential runtime. The original approach has been attibuted to Kraitchik [Kra22, pp. 119–123], [Kra24, pp. 69–70, 216–267] with similar ideas used by Western and Miller [WM68]. Index calculus has been successfully used to factor large composite integers and to compute discrete logarithms in many types of groups. In this section we present a brief overview of index calculus as applied to the discrete logarithm problem. Similar surveys are available in Odlyzko [Odl85, §4], McCurley [McC90, §5], and Schirokauer, Weber and Denny [SWD96, §3].

4.2.1 Index Calculus in Generic Groups

Suppose we wish to solve an instance of the discrete logarithm problem in a multiplicative, cyclic group G of order n_G. That is, we are given two elements $g, h \in G$ such that $h = g^b$ for some $b \in \mathbb{N}$, and we wish to find b. In the following description we assume that g is a generator of the group G, *i.e.* the order $|g| = n_G$. Similar to the framework given by Enge and Gaudry [EG02, §2], we require that there exists a free Abelian monoid M over a countable set $\mathcal{P}$, together with an equivalence relation $\sim$ compatible with the operation on G such that $G \cong M/\sim$. The elements in $\mathcal{P}$ are called *primes* and this setup allows the unique decomposition of the elements of G into the primes of $\mathcal{P}$. The **index calculus algorithm** consists of four major steps:

i) *Choose a factor base.* A **factor base** is the set $\mathcal{F}_B$ consisting of primes $\pi \in \mathcal{P}$ satisfying $\deg \pi < B$ for a degree function $\deg : M \to \mathbb{R}_{\geq 0}$ and some $B \in \mathbb{N}$ called the **smoothness bound**. The choice of B is important in the running time of the algorithm, but we will discuss this later. Let the cardinality $\#\mathcal{F}_B = n_B$. An element of G is called B-**smooth** if it can be expressed as a product of primes from the factor base $\mathcal{F}_B$.

ii) *Generate relations.* Generate a random element $a_k \in G$ by choosing $k \in \mathbb{N}$ at random satisfying $0 < k < n_G$ and computing $a_k = g^k \in G$. Test if a_k is B-smooth. If it is indeed B-smooth, then we have the following relation:

$$a_k = \prod_{j=1}^{n_B} \pi_j^{e_j} \in G, \qquad \text{where each } \pi_j \in \mathcal{F}_B \text{ and } e_j \in \mathbb{Z}.$$

Then we have a vector $\mathbf{e} = (e_1, e_2, \ldots, e_{n_B})$ which we store along with k. Note that with an appropriately chosen B, the vector $\mathbf{e}$ is likely to have only a small

number (in comparison to n_B) of non-zero entries; such a vector is said to be **sparse**. Repeat this step until we have a set $\mathcal{R}_B$ of m relations, for some $m \geq n_B$. Generally, m is just slightly larger than n_B in an effort to ensure there are n_B linearly independent relations. This relation generation step can be done in parallel on multiple processors.

iii) *Solve a linear system.* With the set $\mathcal{R}_B = \{(\mathbf{e}_1, k_1), (\mathbf{e}_2, k_2), \ldots, (\mathbf{e}_m, k_m)\}$ of relations found in Step 2, let $\mathbf{b} = (k_1, \ldots, k_m)$ and construct an $n_B \times m$ matrix $A_{\mathcal{R}}$ consisting of column vectors $\mathbf{e}_1^{\mathrm{tr}}, \mathbf{e}_2^{\mathrm{tr}} \ldots, \mathbf{e}_m^{\mathrm{tr}}$, where $\mathbf{e}_i^{\mathrm{tr}}$ denotes the transpose of the vector $\mathbf{e}_i$. Then one solves the following system:

$$\mathbf{x} A_{\mathcal{R}} \equiv \mathbf{b} \pmod{n_G} . \tag{4.1}$$

If n_B of the relations in $A_{\mathcal{R}}$ are linearly independent, the system is overdetermined and thus solvable. The solution vector $\mathbf{x} = (x_1, \ldots, x_{n_B})$ corresponds to the discrete logarithms of the form $\pi_i = g^{x_i} \in G$ since the i-th entry of $\mathbf{b}$ is computed in Equation (4.1) as

$$k_i = \sum_{\substack{j=1 \\ e_j \in \mathbf{e}_i}}^{n_B} e_j x_j = \sum_{\substack{j=1 \\ e_j \in \mathbf{e}_i}}^{n_B} e_j \log_g \pi_j \pmod{n_G} .$$

Since the matrix A is a sparse matrix, it can be represented using much less memory and there exist efficient methods for computing solutions to the linear system in Equation (4.1). If n_G is not prime, then extra work to solve Equation (4.1) is required (*cf.* Enge and Gaudry [EG02, §4]).

iv) *Extract a solution.* Choose $k \in \mathbb{N}$ at random satisfying $0 < k < n_G$ and compute

$h \cdot g^k \in G$, testing if the result is B-smooth. If it is not B-smooth choose a different k until it is. Then for a vector $\mathbf{d} = (d_1, \ldots, d_{n_B})$ we have the following relation:

$$h \cdot g^k = \prod_{j=1}^{n_B} \pi_j^{d_j}, \qquad \text{where each } \pi_j \in \mathcal{F}_B \text{ and } d_j \in \mathbb{N}_0 .$$

Now we can compute the discrete logarithm $b = \log_g h$ as

$$b = \sum_{i=1}^{n_B} d_i \cdot x_i - k \pmod{n_G} .$$

The choice of the smoothness bound B is important in the runtime of the algorithm. If B is too small, it will take too long to generate enough relations, since finding B-smooth elements will be difficult; conversely, if B is too large, the linear system will be too big to solve efficiently.

Suppose the smoothness bound B is chosen such that $n_B \in O\big(\mathsf{L}_{n_G}(\beta)\big)$ for some constant $\beta > 0$. Let $n_{G/B}$ be the number of B-smooth elements in G. Then, if we test elements of G uniformly at random, we expect to find a B-smooth element after $\frac{n_G}{n_{G/B}}$ tries. We assume that $\frac{n_G}{n_{G/B}} \in O\big(\mathsf{L}_{n_G}(\sigma)\big)$ for some constant $\sigma > 0$. We also assume one can decompose a B-smooth element of G into its prime factors is in $\widetilde{O}(n_B)$, and one can test an element of G for B-smoothness in $\widetilde{O}(n_B^\tau)$ for some constant τ. If generating m relations for some $m \in O(n_B)$ gives a sufficient probability that the relation matrix A has full rank, then using efficient linear algebra techniques, Enge and Gaudry [EG02, §6] show that the number of operations in G required for the index calculus method to solve an instance of the discrete logarithm problem in G is

in

$$O\left(\mathbf{L}_{n_G}\left(\max\left\{1,\ 2\beta,\ (1+\tau)\beta+\sigma\right\}+o(1)\right)\right).$$

4.3 Index Calculus in the Infrastructure

Müller, Stein and Thiel were the first to describe an index calculus variant in the infrastructure for fields of odd characteristic [MST99]. In this section we expand upon the MST algorithm, providing a new relation generation method, an updated complexity analysis, and a generalization to both even and odd characteristics.

Let $K = \mathbb{F}_q$ be a finite field of order q. Let $C : y^2+h(x)y = f(x)$ be a hyperelliptic curve of genus g, where $f, h \in K[x]$. If $\operatorname{char} K > 2$, then we assume that $\deg f = 2g+2$ and $\operatorname{sgn} f$ is a square in K; otherwise, if $\operatorname{char} K = 2$ (*i.e.* $q = 2^m$, for some $m \in \mathbb{N}$), then we assume that $\deg h = g+1$ and either $\deg f \leq 2g+1$ or $\deg f = 2g+2$ and $\operatorname{sgn} f = e^2+e$ for some $e \in K^*$. Then from Section 3.1 we have that $F = K(C)$ is a real quadratic function field. Let $\mathcal{O}$ denote the coordinate ring $K[C]$ and let $R_{\mathcal{O}}$ be the regulator of $\mathcal{O}$.

4.3.1 Overview

We want to apply the framework of the index calculus method in the previous section to solve the computational problems given in Section 4.1 in the setting of the infrastructure of a real quadratic function field. However, there are a number of changes that must be made to the framework in order to apply to our setting.

Since f and h are polynomials in $K[x]$ of degree $O(g)$, we can describe the function

field by giving the equation C in $O(g \log q)$ bits. For an instance of the infrastructure discrete logarithm problem as given in Problem 4.6, the reduced $\mathcal{O}$-ideal $\mathfrak{b}$ given in standard representation can also be expressed in $O(g \log q)$ bits. Therefore, we consider the input size to our index calculus algorithm to be $g \log q$ bits.

Recall that the infrastructure is the cycle of reduced ideals in an ideal class of $\mathrm{Cl}^0(\mathcal{O})$. We will work in the principal ideal class. The infrastructure operation $\circledast$ is based on the notion of distance in the cycle, since the result is the reduced principal $\mathcal{O}$-ideal with distance closest to the sum of the distances of the two operands. However, because we may need to correct the ideal from the operation $*$ to satisfy our previous property, the infrastructure operation $*$ is not associative; hence, the infrastructure is not a group under $*$ as was assumed in our index calculus framework.

We do have a notion of prime ideals in $\mathcal{O}$, so we are still able to compute a factor base. Similar to the previous framework, our relations consist of a vector $\mathbf{e}$ indicating how a B-smooth element factors over the factor base. In the infrastructure, this B-smooth element is a reduced $\mathcal{O}$-ideal $\mathfrak{a}$, and $\mathbf{e}$ contains the exponents for the prime ideals in the factor base to obtain $\mathfrak{a}$. However, unlike in the previous section where we solved the linear system to obtain the discrete logarithms of the smooth ideals, we keep track of the distances of the smooth ideals and include that information in the relation. That is, relations consist of the vector $\mathbf{e}$ indicating how $\mathfrak{a}$ factors, along with the distance $\delta(\mathfrak{a})$.

Similar to our framework, we put each vector from the relation in a column of a matrix A. If we are computing an infrastructure DLP, *i.e.* trying to find $\delta(\mathfrak{b})$ for some principal $\mathcal{O}$-ideal $\mathfrak{b}$, then we first find an equivalent smooth ideal $\mathfrak{b}'$ with known distance $\delta(\mathfrak{b}', \mathfrak{b})$. Then we let $\mathbf{b}$ be the vector indicating how $\mathfrak{b}'$ factors over the prime

ideals in the factor base. We solve the linear system $A\mathbf{x} = \mathbf{b}$. In this case, the solution vector $\mathbf{x}$ gives a linear combination of the relation vectors that results in $\mathbf{b}$. That is, $\mathbf{x}$ indicates how to apply the infrastructure operation $\circledast$ to get $\mathfrak{b}'$ from the ideals of the relations. Therefore, we can use the distance property of $\circledast$ and compute the dot product of $\mathbf{x}$ with the distances of the relations to get the distance $\delta(\mathfrak{b}'')$, where $\delta(\mathfrak{b}'') \equiv \delta(\mathfrak{b}') \bmod R_{\mathcal{O}}$.

To compute the regulator $R_{\mathcal{O}}$, one notes that if we let the vector $\mathbf{b} = (0, 0, \ldots, 0)$, this vector corresponds to a relation for the ideal $\mathcal{O}$. Therefore, solving for a kernel vector of A gives a vector $\mathbf{x}$ that is a linear combination of the relations to the $\mathcal{O}$-relation. Hence, the dot product of $\mathbf{x}$ with the distances of the relations gives the distance of some multiple of the regulator $R_{\mathcal{O}}$. We can find the actual regulator by taking the smallest divisor d that satisfies $\rho^*(\mathcal{O}, d) = (\mathcal{O}, \delta)$ with $\delta > 0$.

Alternatively, if we want both the regulator $R_{\mathcal{O}}$ and the class number $h_{\mathcal{O}}$, we can form an augmented matrix A' that consists of A with the distances added as a row on the bottom. If A' has full rank, this augmented matrix corresponds to basis for a sublattice of the lattice of all possible relations. It is known that this lattice has determinant $h_{\mathcal{O}}R_{\mathcal{O}}$, so the Hermite normal form of A' has a determinant equal to a multiple of $h_{\mathcal{O}}R_{\mathcal{O}}$. If A' is a basis for the full relation lattice, then the determinant is exactly $h_{\mathcal{O}}R_{\mathcal{O}}$.

Finally, to compute the class group structure or just the class number $h_{\mathcal{O}}$, one computes the Smith normal form of the HNF of A. The invariant factors of the SNF that are greater than one correspond to the group structure. The product of the invariant factors gives the ideal class number.

We explain each of these stages in detail in the following sections.

4.3.2 Computing the Factor Base

Since $\mathcal{O}$ is a Dedekind domain, every $\mathcal{O}$-ideal is a unique product of prime $\mathcal{O}$-ideals. Then using the fact that the generalized Riemann hypothesis (GRH) is proven to be true in algebraic function fields,[3] the following result gives a bound on the size of elements to be included in the factor base [MST99, §3]:

Theorem 4.7 (Müller, Stein & Thiel, 1999). *The ideal class group* $\mathrm{Cl}(\mathcal{O})$ *is generated by the prime ideals* $\mathfrak{p}$ *with norm satisfying* $\deg N(\mathfrak{p}) \leq B$ *if the smoothness bound satisfies*

$$B \geq \left\lceil 2\log_q(4g-2) \right\rceil .$$

Given this result, we define the **degree** of a reduced $\mathcal{O}$-ideal $\mathfrak{a}$ to be $\deg \mathfrak{a} = \deg N(\mathfrak{a})$, where the norm is the polynomial $a \in K[x]$ from $\mathfrak{a} = (a, b)$ in standard representation.

We select our factor base $\mathcal{F}_B$ to contain the prime $\mathcal{O}$-ideals of degree at most B that lie above either a splitting or ramified $K[x]$-ideal. Such prime $\mathcal{O}$-ideals are of the form $\mathfrak{P} = (a, b)$, where $a = p$ is a monic irreducible polynomial in $K[x]$ and b is a root of $\Psi(y) = y^2 + hy - f$. Note that if K has order q, then there are q^d monic polynomials of degree d in $K[x]$. If we determine that a monic polynomial $p \in K[x]$ with $\deg p \leq B$ is irreducible, then we determine the splitting behaviour of $\mathfrak{p} = pK[x]$ based on how $\Psi(y) = y^2 + hy - f$ factors in $(K[x]/\mathfrak{p})[y]$. We have that $\mathfrak{p}$ is ramified if $\Psi(y)$ is a square. The ideal $\mathfrak{p}$ splits in $\mathcal{O}$ if $\Psi(y)$ has two unique roots. Finally, $\Psi(y)$ is inert if $\Psi(y)$ has no roots.

[3] The proof of the GRH in algebraic function fields in the case of genus 1 is due to Hasse [Has36], and proven for arbitrary genus by Weil [Wei48].

In the odd characteristic case we have $h = 0$, so if $p|f$, then $\Psi(y)$ is a square and we have that $\mathfrak{p}$ is ramified below the prime $\mathcal{O}$-ideal $\mathfrak{P} = (p, 0)$. Otherwise, if $p \nmid f$, then $\mathfrak{p}$ splits if and only if f is a square in $K[x]/\mathfrak{p}$. Using an algorithm that finds the squarefree decomposition of a polynomial, we can determine whether f is a square modulo p (*cf.* Bach and Shallit [BS96, p. 169]). If we determine that f is a square, the square root $b = \sqrt{f} \bmod p$ can be probabilistically computed using either the algorithm of Tonelli-Shanks[4] or Cipolla-Lehmer [BS96, pp. 155–159]. Once we have computed b, we know that $\mathfrak{p}$ splits into a product of two conjugate prime $\mathcal{O}$-ideals $\mathfrak{P} = (p, b)$ and $\overline{\mathfrak{P}} = (p, -b)$.

In the even characteristic case where $|K| = 2^n$, we can determine if $\Psi(y) = y^2 + hy - f$ is a square by checking if $p|h$. This works since $\frac{\mathrm{d}}{\mathrm{d}y}\Psi = 2y + h \equiv 0 \bmod p$ if and only if $p|h$. If Ψ is a square, then a square root can be computed as $b = \Psi^{2^{n+\deg p-1}} \bmod p$ and $\mathfrak{p}$ is ramified below the prime ideal $\mathfrak{P} = (p, b)$ [BS96, p. 155]. For a non-square $\Psi(y)$, $\mathfrak{p}$ splits if and only if $\Psi(y)$ has roots in $K[x]/\mathfrak{p}$. If $\deg p = m$, then $K[x]/\mathfrak{p} \cong \mathbb{F}_{2^{nm}}$. The **trace** of an element $a \in \mathbb{F}_{2^{nm}}$ is a map defined as

$$\mathrm{Tr} : \begin{cases} \mathbb{F}_{2^{nm}} & \to & \mathbb{F}_2 \\ a & \mapsto & \sum_{i=0}^{nm-1} a^{2^i} \end{cases}.$$

Consider the change of variables $u = yh^{-1} \bmod p$ on $\Psi(y)$ and let $a = fh^{-2} \bmod p$, giving the Artin-Schreier polynomial $t(u) = u^2 + u - a$. Then it follows from Hilbert's Theorem 90 that $t(u)$ has a root in $K[x]/\mathfrak{p}$ if and only if $\mathrm{Tr}(a) = 0$ [Sti93, p. 241]. Once we know $t(u)$ has roots, one of them can be found using the McEliece polynomial factorization algorithm, a randomized Las Vegas algorithm that uses the trace to find

[4]Shanks called this algorithm RESSOL and it is sometimes referred to in the literature as such [Sha72a, §5].

a non-trival factor with a failure probability of at most 1/2 [McE69, §II]. If d is a root of $t(u)$, then $b = hd$ is a root of $\Psi(y)$.[5] Consequently, $\mathfrak{p}$ will split into the two conjugate prime $\mathcal{O}$-ideals $\mathfrak{P} = (p, b)$ and $\overline{\mathfrak{P}} = (p, -b - h)$.

For each splitting ideal $\mathfrak{p} = pK[x]$ such that $\mathfrak{p}\mathcal{O} = \mathfrak{P}\overline{\mathfrak{P}}$, we include only one of $\mathfrak{P}, \overline{\mathfrak{P}}$ in $\mathcal{F}_B$. The complete procedure can be seen in Algorithm 4.8. To have $\mathcal{F}_B$ satisfy Theorem 4.7, Müller, Stein and Thiel estimate for the number of prime ideals that will be in $\mathcal{F}_B$ is bounded as $n_B \leq 4Bq^B$ [MST99, §2.2].

Algorithm 4.8 (Factor base generation). *Generates the factor base for index calculus in the infrastructure.*

Input: A smoothness bound B for a hyperelliptic curve $C : y^2 + h(x)y = f(x)$ over a field $K = \mathbb{F}_q$

Output: A factor base $\mathcal{F}_B$ that generates the ideal class group $\mathrm{Cl}(\mathcal{O})$

1: $\mathcal{F}_B \leftarrow \{\emptyset\}$, $m \leftarrow 1$
2: **repeat**
3: **for each** monic polynomial $a \in K[x]$ with $\deg a = m$ **do**
4: **if** a is irreducible over $K[x]$ **then**
5: $b \leftarrow y^2 + hy - f$
6: **if** $h = 0 \ \wedge \ a | f$ **then** ▷ Odd characteristic cases
7: $b \leftarrow 0$ ▷ $aK[x]$ is ramified
8: **else if** $h = 0 \ \wedge$ Squarefree$(f \bmod a) = 1$ **then**
9: $b \leftarrow \sqrt{f} \pmod{a}$ ▷ $aK[x]$ splits
10: **else if** $h \neq 0 \ \wedge \ a | h$ **then** ▷ Even characteristic cases
11: $b \leftarrow (y^2 - f)^{2^{\deg a - 1}q} \pmod{a}$ ▷ $aK[x]$ is ramified

[5] Equivalently, we could just as easily use the McEliece algorithm to find a non-trival factor of $\Psi(y) \bmod p$ directly.

12: **else if** $h \neq 0 \ \wedge \ \mathrm{Tr}(fh^{-2}) = 0$ **then**

13: $(y - b) \leftarrow \mathsf{PolyFactor}(y^2 + hy - f \bmod a)$ ▷ $aK[x]$ splits

14: **if** $b \neq y^2 + hy - f$ **then** ▷ If $aK[x]$ is inert, $b = y^2 + hy - f$

15: $\mathcal{F}_B \leftarrow \mathcal{F}_B \cup \{(a, b)\}$

16: $m \leftarrow m + 1$

17: **until** $m > B$

Let $L(n) = \log n \log\log n$. Ben-Or's algorithm for testing a degree-n polynomial in $K[x]$ for irreducibility requires $O(n^2 L(n) \log n \log q)$ operations in K in the worst case, but only $O(nL(n) \log n \log q)$ operations on average [Ben81, §2]. To iterate over every monic polynomial $a \in K[x]$, we need to perform the irreducibility test $T = \sum_{k=1}^{B} q^k \in O(q^B)$ times. Dividing a degree-m polynomial by a degree-n polynomial requires $O(n(m - n + 1))$ operations in K [Knu97, pp. 420–421]. Tonelli-Shanks runs in expected $O(n \log^4 q)$ bit operations and Cipolla-Lehmer is in $O(n \log^3 q)$ bit operations, where $n = \deg a$.[6] We can find a non-trivial factor of a quadratic polynomial in $K[x, y]/aK[x]$ when $\mathrm{char}\, K = 2$ using the McEliece algorithm in expected $O(\log^3 q)$ bit operations with a failure probability of at most $1/2$ [BS96, pp. 155–159, 355]. Since $\deg f \in O(g)$ and $\deg a \in O(B)$, we have the expected number of field operations required to perform Algorithm 4.8 is in

$$O\Big(TB \cdot L(B) \log B \log q + n_B\big(gB + B \log^3 q\big)\Big)$$
$$\subseteq O\big(q^B B \cdot L(B) \log B \log q + q^B B^2 g + q^B B^2 \log^3 q\big)\big) \ . \tag{4.2}$$

[6] Although Cipolla-Lehmer has a faster asymptotic upper bound for its runtime, in practice the runtime of Tonelli-Shanks is comparable unless $q - 1$ is highly divisible by 2 (*cf.* Bach and Shallit [BS96, pp. 158–159]).

We continue the analysis following the work of Enge [Eng02]. Suppose we choose the smoothness bound $B = \lceil \log_q \mathbf{L}_{q^g}(c) \rceil$ for some constant $c \in \mathbb{R}_{>0}$ satisfying Theorem 4.7. Since $\log_q \mathbf{L}_{q^g}(c)$ is rounded up to B, we have q^B potentially almost as big as $q^{\log_q \mathbf{L}_{q^g}(c)+1} = q \cdot \mathbf{L}_{q^g}(c)$. Then to remain subexponential we require q to be subexponential in $O(g \log q)$. Therefore, the genus g must be large. Suppose $g \geq \vartheta \log q$ for some constant $\vartheta > 0$, then

$$q = e^{\log q} = \exp\left(\tfrac{1}{\sqrt{\vartheta}}\sqrt{\vartheta(\log q)^2}\right) \leq \exp\left(\tfrac{1}{\sqrt{\vartheta}}\sqrt{g \log q}\right) \leq \mathbf{L}_{q^g}\left(\tfrac{1}{\sqrt{\vartheta}}\right) .$$

Now we have the following upper bound that is satisfied even if $B \approx \log_q \mathbf{L}_{q^g}(c) + 1$:

$$q^B \leq \mathbf{L}_{q^g}(c) \cdot \mathbf{L}_{q^g}\left(\frac{1}{\sqrt{\vartheta}}\right) = \mathbf{L}_{q^g}\left(c + \frac{1}{\sqrt{\vartheta}}\right) . \tag{4.3}$$

Continuing from Equation (4.2), the expected number of field operations required for Algorithm 4.8 to generate the factor base $\mathcal{F}_B$ is in

$$\begin{aligned} &\widetilde{O}\left(\mathbf{L}_{q^g}\left(c + \frac{1}{\sqrt{\vartheta}}\right) \log q + \mathbf{L}_{q^g}\left(c + \frac{1}{\sqrt{\vartheta}}\right) g + \mathbf{L}_{q^g}\left(c + \frac{1}{\sqrt{\vartheta}}\right) \log^3 q\right) \\ &\subseteq O\left(\mathbf{L}_{q^g}\left(c + \frac{1}{\sqrt{\vartheta}} + o(1)\right)\right) , \end{aligned} \tag{4.4}$$

when $g \log q \to \infty$ for $0 < \vartheta \leq g/\log q$.

Comparison with Previous Work

Müller, Stein and Thiel obtained a runtime of $O(Bq^B(\deg f)^3 \log q)$ operations in odd characteristic fields to compute the factor base [MST99, §5]. This simplifies using our

analysis as

$$\widetilde{O}\left(g^3 \log q \cdot \mathbf{L}_{q^g}\left(c+\frac{1}{\sqrt{\vartheta}}\right)\right) \subseteq O\left(\mathbf{L}_{q^g}\left(c+\frac{1}{\sqrt{\vartheta}}+o(1)\right)\right).$$

Note that the subexponential term has swallowed a much larger polynomial function than in Equation (4.4).

Enge's asymptotic complexity for computing the factor base in the Jacobian [Eng02, §5.3] is

$$\widetilde{O}\left(\log^2 q \cdot \mathbf{L}_{g^q}\left(2c+\frac{1}{\sqrt{\vartheta}}\right)\right) \subseteq O\left(\mathbf{L}_{g^q}\left(2c+\frac{1}{\sqrt{\vartheta}}+o(1)\right)\right).$$

Enge has a larger multiple on the constant c than our result in Equation (4.4), since his algorithm uses trial division to determine irreducible polynomials.

4.3.3 Smoothness Testing

Suppose we are given a reduced $\mathcal{O}$-ideal $\mathfrak{a} = (a_\mathfrak{a}, b_\mathfrak{a})$ in standard representation, and we wish to determine whether $\mathfrak{a}$ corresponds to a B-smooth ideal. We will see that we can perform an efficient test using the norm $\mathrm{N}(\mathfrak{a}) = a_\mathfrak{a}$.

Let the prime ideals in the factor base be labelled as follows: $\mathcal{F}_B = \{\mathfrak{p}_1, \mathfrak{p}_2, \ldots, \mathfrak{p}_{n_B}\}$. Each prime ideal $\mathfrak{p}_i = (a_{\mathfrak{p}_i}, b_{\mathfrak{p}_i})$ in $\mathcal{F}_B$ has $\mathrm{N}(\mathfrak{p}_i) = a_{\mathfrak{p}_i}$ irreducible in $K[x]$. Suppose the norm of $\mathfrak{a}$ factors over the norms of the $\mathfrak{p}_i \in \mathcal{F}_B$. Then there exist exponents $e_i \in \mathbb{N}_0$ such that

$$\mathrm{N}(\mathfrak{a}) = \prod_{\mathfrak{p}_i \in \mathcal{F}_B} \mathrm{N}(\mathfrak{p}_i)^{e_i} = \prod_{(a_{\mathfrak{p}_i}, b_{\mathfrak{p}_i}) \in \mathcal{F}_B} a_{\mathfrak{p}_i}^{e_i}. \tag{4.5}$$

The factorization the norm of an ideal $\mathfrak{a}$ may not exactly correspond to the factorization of the ideal $\mathfrak{a}$. There are two reasons for this. First, we must take into account $b_{\mathfrak{a}}$ from the standard representation $\mathfrak{a} = (a_{\mathfrak{a}}, b_{\mathfrak{a}})$. Therefore, we must correct the sign of the exponents to get the ideal $\mathfrak{a}$ with the correct conjugation. This is done as follows: for each i, $1 \leq i \leq n_B$, find $s_i \in \{-1, 1\}$ satisfying $b_{\mathfrak{a}} \equiv s_i \cdot b_{\mathfrak{p}_i} \pmod{a_{\mathfrak{p}_i}}$. This gives a vector $\mathbf{e} = (s_1 e_1, s_2 e_2, \ldots, s_{n_B} e_{n_B})$. The second reason why the factorization of the norm of $\mathfrak{a}$ does not directly result in $\mathfrak{a}$ being B-smooth is because the negative signs in $\mathbf{e}$ are ideal inverses and result in a polynomial coefficient for the ideal. According to Equation (3.3), we have

$$\prod_{i=1}^{n_B} \mathfrak{p}_i^{s_i \cdot e_i} = \mathfrak{a} \prod_{i:\substack{e_i \in \mathbf{e} \\ s_i < 0}} \left(\frac{1}{\mathrm{N}(\mathfrak{p}_i)}\right)^{|e_i|} = \mathfrak{a} \prod_{i:\substack{e_i \in \mathbf{e} \\ s_i < 0}} \frac{1}{a_{\mathfrak{p}_i}^{|e_i|}}. \tag{4.6}$$

Therefore, for each reduced $\mathcal{O}$-ideal $\mathfrak{a} = (a_{\mathfrak{a}}, b_{\mathfrak{a}})$ whose norm factors as in Equation (4.5), we have a B-smooth ideal with the standard representation $(s, a_{\mathfrak{a}}, b_{\mathfrak{a}})$, where $s = \prod_{e_i \in \mathbf{e}, s_i < 0} a_{\mathfrak{p}_i}^{-e_i}$ and $s_i \equiv b_{\mathfrak{a}} b_{\mathfrak{p}_i}^{-1} \pmod{a_{\mathfrak{p}_i}}$.

To efficiently test whether the norm of a reduced $\mathcal{O}$-ideal $\mathfrak{a}$ factors according to Equation (4.5), we use a strategy called **distinct degree factorization**. This is based on the fact that if the field K has order q, then $x^{q^B} - x$ is equal to the product of all monic irreducible polynomials in $K[x]$ of degree dividing B. Then if $\mathrm{N}(\mathfrak{a}) = a_{\mathfrak{a}}$ is squarefree, the product of the degree-1 factors of a_{a_j} is given by $d_1 = \gcd(x^q - x, a_{\mathfrak{a}})$. This can be iterated since the product of the degree-2 factors of $a_{\mathfrak{a}}$ is given by $d_2 = \gcd(x^{q^2} - x, a_{\mathfrak{a}}/d_1)$ [BS96, pp. 170–171]. In fact, instead of computing the GCDs for degrees 1 to B, using an optimization described by Velichka, we can reduce this to only compute the GCDs for degrees $\lfloor \frac{t}{2} \rfloor + 1$ to B [Vel08, p. 42].

This is sufficient since degrees of $\lfloor \frac{t}{2} \rfloor$ and below must divide some degree in the higher range.

To efficiently compute $x^{q^i} \bmod a_a$ for $1 \leq i \leq B$, we make use of special map called the **Frobenius endomorphism**. Let $q = p^m$ with p prime and $m \in \mathbb{N}$. Then the Frobenius endomorphism on $K = \mathbb{F}_q$ is the automorphism defined by $\phi : a \mapsto a^p$. If $m = 1$, then ϕ is just the identity automorphism due to Fermat's little theorem. The Frobenius endomorphism can be iterated as $\phi^2(a) = \phi\big(\phi(a)\big) = a^{p^2}$ to get the **n-th power Frobenius map** $\phi^n : a \mapsto a^{p^n}$ for any $n \in \mathbb{N}$. Note that for any $a \in K$ the result of the m-th power Frobenius map is $\phi^m(a) = a^q$ [BS96, pp. 133–134]. We will refer to the latter simply as the **Frobenius map**, denoted by $\Phi(a) = \phi^m(a)$.

One nice property of the Frobenius map is that $\Phi(a) = a$ if and only if $a \in \mathbb{F}_p$. Another property that $\Phi^{i+j} = \Phi^i \circ \Phi^j$ gives us a computational advantage when iterating the Frobenius map with $m > 1$ or in a field extension of K. Consider the field extension $L \supset K$ where $L = K[x]/a_a K[x]$. We can use the binary expansion of $n = \sum_{k=0}^{\ell} 2^{b_k}$ with $\ell = \lfloor \log n \rfloor$ and $b_k \in \{0, 1\}$ to compute $\Phi^n(x) = x^{q^n} \bmod a_a$ using the square-and-multiply method in Algorithm 4.9 [vzGG03, pp. 388–390].

Algorithm 4.9 (Iterative Frobenius map Φ^n). *Compute $\Phi^n(a) \bmod t$ for some n and polynomial $a \in K[x]/tK[x]$.*

Input: An integer $n \in \mathbb{N}$ and its binary representation $n = \sum_{k=0}^{\ell} 2^{b_k}$ for $\ell = \lfloor \log n \rfloor$; a polynomial $a \in L$, where $L = K[x]/tK[x]$ for a field K of order q and a monic irreducible polynomial $t \in K$ of degree m.

Output: The polynomial $c = \Phi^n(a) \in L$.

1: $s \leftarrow x^q \pmod t$, $c \leftarrow 1$

2: **for** i **from** 0 **to** ℓ **do**

3: **if** $b_i = 1$ **then**

4: $\quad c \leftarrow c \cdot s \pmod{t}$

5: $s \leftarrow s(s) \bmod t$ ▷ Evaluate the polynomial $s(x)$ at s

6: **if** $\deg c > 0$ **then**

7: $\quad c \leftarrow c(a) \bmod t$ ▷ Evaluate the polynomial $c(x)$ at a

Step 1 of Algorithm 4.9 can be computed in $O(mL(m)\log q)$ operations in K, where $L(m) = \log m \log\log m$, using the square-and-multiply algorithm in the extension field L. We can evaluate a polynomial in L in $O(m)$ operations in K. Then the entire runtime of Algorithm 4.9 is in $O(mL(m)\log q)$.

Now we return our focus to the smoothness test. If we have determined that there is a B-smooth ideal corresponding to $\mathfrak{a}$ via distinct degree factorization, we complete the factorization of $\mathrm{N}(\mathfrak{a})$ using a polynomial factoring algorithm and place the multiplicities of the factors in the appropriate entries of the vector $\mathbf{e}$. The complete method for smoothness testing and determining the factorization is presented in Algorithm 4.10 and we denote it by σ.

Algorithm 4.10 (Smoothness test σ). *Test whether a reduced ideal is B-smooth with a possible coefficient and return the relation vector if it is.*

Input: A factor base $\mathcal{F}_B = \{\mathfrak{p}_1 = (a_{\mathfrak{p}_1}, b_{\mathfrak{p}_1}), \ldots, \mathfrak{p}_{n_B} = (a_{\mathfrak{p}_{n_B}}, b_{\mathfrak{p}_{n_B}})\}$ and a reduced $\mathcal{O}$-ideal $\mathfrak{a} = (a_\mathfrak{a}, b_\mathfrak{a})$.

Output: A vector $\mathbf{e} = (e_1, \ldots, e_{n_B})$ such that $\prod_{i=1}^{n_B} \mathfrak{p}_i^{e_i}$ if there is a B-smooth ideal $(s, a_\mathfrak{a}, b_\mathfrak{a})$ for some coefficient $s \in K(x)$; or the zero vector otherwise.

1: $a \leftarrow \mathsf{Squarefree}(a_\mathfrak{a})$ ▷ Compute the squarefree part of $a_\mathfrak{a}$

2: $\xi \leftarrow \Phi^{\lfloor \frac{t}{2} \rfloor}(x) \pmod{a}$ ▷ Iterative Frobenius map (Alg. 4.9)

3: **for** i **from** $\lfloor \frac{t}{2} \rfloor + 1$ **to** B **do** ▷ Perform distinct degree factorization
4: $\quad \xi \leftarrow \Phi(\xi) \pmod{a}$ ▷ Frobenius map
5: $\quad d \leftarrow \gcd(\xi - x, a)$
6: $\quad a \leftarrow a/d$
7: **if** $a \neq 1$ **then** ▷ $\mathfrak{a}$ is not B-smooth
8: $\quad \mathbf{e} \leftarrow (0, \ldots, 0)$
9: **else** ▷ $\mathfrak{a}$ must be B-smooth
10: $\quad p_1^{n_1} p_2^{n_2} \cdots p_\ell^{n_\ell} \leftarrow \mathsf{PolyFactor}(a_{\mathfrak{a}})$ ▷ Compute the factors of $a_{\mathfrak{a}}$
11: $\quad e_1, e_2, \ldots, e_{n_B} \leftarrow 0$ ▷ Initialize the e_i to 0
12: $\quad$ **for** i **from** 1 **to** ℓ **do**
13: $\quad\quad$ Find k such that $p_i = a_{\mathfrak{p}_k}$ for $\mathfrak{p}_k \in \mathcal{F}_B$
14: $\quad\quad e_k \leftarrow n_i$
15: $\quad\quad$ **if** $b_{\mathfrak{a}} \not\equiv b_{\mathfrak{p}_k} \pmod{p_i}$ **then**
16: $\quad\quad\quad e_k \leftarrow -e_k$ ▷ Correct the sign of the exponent
17: $\quad \mathbf{e} \leftarrow (e_1, e_2, \ldots, e_{n_B})$

An algorithm given by Yun [Yun77] for computing the squarefree decomposition of a degree n polynomial in $K[x]$, where K has order q, requires $O\big(((1+n)\log q)^2\big)$ bit operations [BS96, pp. 170, 356–357]. The distinct degree factorization requires $O\big((n+\log q)((1+n)\log q)^2\big)$ bit operations [BS96, p. 171]. Cantor and Zassenhaus [CZ81] gave a Las Vegas algorithm to find a non-trivial factor of a polynomial in an expected running time of $O\big((n+\log q)((1+n)\log q)^2\big)$ with a failure probability of at most 2^{1-r}, where r is the number of irreducible factors [BS96, p. 167]. It can be repeated to find the complete factorization of a polynomial. If one does the searching for k in a reasonable way, we expect that the polynomial factorization will dominate

the running time of the algorithm. Since the ideals we are testing are always reduced, we have $n \in O(g)$ giving a total running time for Algorithm 4.10 in

$$O\big((g + \log q)((1 + g)\log q)^2\big) \subseteq O(g^3 \log^2 q + g^2 \log^3 q) .$$

4.3.4 Generating Relations via a Baby Step Walk

In this section we describe how to generate a set $\mathcal{R}_B$ of relation vectors from which we will form a relation matrix $A_{\mathcal{R}}$. Müller, Stein and Thiel [MST99] generate relations in the infrastructure following a method introduced by Hafner and McCurley [HM89] and generalized by Buchmann [Buc90]. However, their method is slower and results in the relation matrix $A_{\mathcal{R}}$ being dense. To get sparse relations we propose an alternate method which we call a **baby walk**. The baby walk simply performs baby steps from an $\mathcal{O}$-ideal $\mathfrak{a}_0$ to produce a sequence $\mathfrak{a}_j = \rho(\mathfrak{a}_{j-1})$ for $j = 1, 2, \ldots$. If we obtain a B-smooth ideal from $\mathfrak{a}_j$, then the vector $\mathbf{e}$ containing the factorization of the B-smooth ideal and its distance is a relation. As we will see, this not only results in the relation matrix $A_{\mathcal{R}}$ being sparse, but can also be made to ensure that $A_{\mathcal{R}}$ will be nonsingular.

We use Algorithm 4.10 to test each ideal $\mathfrak{a}_j$ in the baby walk for smoothness. The probability that the baby walk efficiently finds B-smooth ideals relies on the following heuristic.

Heuristic 4.11. *The B-smooth $\mathcal{O}$-ideals are evenly distributed within and between the ideal classes of* $\mathrm{Cl}(\mathcal{O})$.

The experiments in Section 5.4.2 suggest that Heuristic 4.11 is true and we will assume it in our analysis.

Once we have found a B-smooth $\mathcal{O}$-ideal $\mathfrak{a}'_j = (s, a_a, b_a)$ with the exponent

vector $\mathbf{e}_j = (e_1, e_2, \ldots, e_{n_B})$ (with corrected signs), we have to compute the distance $\delta(\mathfrak{a}'_j)$. If $\mathfrak{a}_j = (a_a, b_a)$ is the reduced $\mathcal{O}$-ideal from the baby walk satisfying $\mathrm{N}(\mathfrak{a}_j) = \prod_{\mathfrak{p}_i \in \mathcal{F}_B} \mathrm{N}(\mathfrak{p}_i)^{|e_i|}$, then we have the distance $\delta(\mathfrak{a}_j)$ from Algorithm 3.13 and it follows from Equation (4.6) that the distance of the B-smooth ideal is

$$\delta(\mathfrak{a}'_j) = \delta(\mathfrak{a}_j) - \sum_{i:\substack{e_i \in \mathbf{e}_j \\ e_i < 0}} |e_i| \cdot \deg \mathrm{N}(\mathfrak{p}_i) \, .$$

Now we have a relation of the form:

$$\mathfrak{a}'_j = \prod_{i=1}^{n_B} \mathfrak{p}_i^{e_i} = \alpha\mathcal{O} \qquad \text{for some } \alpha \in F \text{ satisfying} \qquad \delta(\mathfrak{a}'_j) = \deg \alpha \, .$$

Therefore, we store $\mathbf{e}_j$ along with $\delta(\mathfrak{a}'_j)$ as a relation.

Suppose we have obtained a set of relations $\mathcal{R}_B = \{(\mathbf{e}_1, \delta_1), (\mathbf{e}_2, \delta_2), \ldots, (\mathbf{e}_{n_B}, \delta_{n_B})\}$ corresponding to n_B B-smooth ideals. Consider the $n_B \times n_B$ square matrix $A_{\mathcal{R}}$ constructed with column vectors $\mathbf{e}_1^{\mathrm{tr}}, \mathbf{e}_2^{\mathrm{tr}}, \ldots, \mathbf{e}_{n_B}^{\mathrm{tr}}$, where $\mathbf{e}_i^{\mathrm{tr}}$ denotes the transpose of vector $\mathbf{e}_i$. We will generate our relations in a similar way as Seysen [Sey87, §4] such that $A_{\mathcal{R}}$ has entries a_{ij}, for $1 \leq i, j \leq n_B$, satisfying

$$|a_{jj}| > \sum_{\substack{i=1 \\ i \neq j}}^{n_B} |a_{ij}| \, .$$

That is, the diagonal entries of $A_{\mathcal{R}}$ are larger than the sum of the other entries in their respective columns. Such a matrix is said to be **strictly diagonally dominant**. It follows that any strictly diagonally dominant matrix is nonsingular [BV07, p. 295].

To obtain a strictly diagonally dominant relation matrix, we require each relation vector $\mathbf{e}_k = (e_1, e_2, \ldots, e_{n_B})$ in $\mathcal{R}_B$ for $1 \leq k \leq n_B$ to have $|e_k| > \sum_{i=1, i \neq k}^{n_B} |e_i|$. Recall

that a reduced $\mathcal{O}$-ideal $\mathfrak{a} = (a_\mathfrak{a}, b_\mathfrak{a})$ has $\deg a_\mathfrak{a} = \deg \mathrm{N}(\mathfrak{a}) \leq g$, where g is the genus of the function field $F = \mathrm{Quot}(\mathcal{O})$. Then if $\mathfrak{a}$ is B-smooth, the norm $\mathrm{N}(\mathfrak{a})$ will factor with exponents $\mathbf{e} = (e_1, e_2, \ldots, e_{n_B})$ such that $\sum_{i=1}^{n_B} |e_i| \leq g$. Therefore, we need each relation vector $\mathbf{e}_k$ to have $|e_k| > g$, for $1 \leq k \leq n_B$.

To generate the k-th relation, $k \leq n_B$, with $|e_k| > g$ using the baby walk, we choose $(\mathfrak{a}_0, \delta_0) = \rho^n\big((\overline{\mathfrak{p}_k})^{2g+1}, (2g+1)\deg \mathrm{N}(\mathfrak{p}_k)\big)$, where $n \in \mathbb{N}_0$ is sufficient to make $\mathfrak{a}_0$ reduced. The baby walk will compute the sequence $\mathfrak{a}_j = \rho(\mathfrak{a}_{j-1})$ for $j = 1, 2, \ldots$ until $\mathfrak{a}_j = (a_{\mathfrak{a}_j}, b_{\mathfrak{a}_j})$ gives us a B-smooth ideal $\mathfrak{a}'_j = (s, a_{\mathfrak{a}_j}, b_{\mathfrak{a}_j})$. While $\mathfrak{a}'_j$ is not necessarily principal, we have some $\gamma \in F$ such that $\mathfrak{a}'_j = (\gamma)\mathfrak{p}_k^{-2g-1}$. After obtaining a vector $\mathbf{e}_k = (e_1, e_2, \ldots, e_{n_B})$ from Algorithm 4.10 corresponding to the factorization of $\mathfrak{a}'_j$, we will have a factorization for the principal ideal $\gamma\mathcal{O}$ as

$$\gamma\mathcal{O} = \mathfrak{p}_k^{2g+1}\mathfrak{a}'_j = \mathfrak{p}_k^{2g+1}\prod_{i=1}^{n_B}\mathfrak{p}_i^{e_i}\,.$$

Therefore, the principal ideal $\gamma\mathcal{O}$ is B-smooth with the relation vector $\mathbf{e}_k = (e_1, \ldots, e_k + 2g+1, \ldots, e_{n_B})$ whose k-th entry will be larger than g. Note that we do not actually compute γ, but we can compute the distance $\delta(\gamma\mathcal{O}) = \deg\gamma$ as

$$\delta(\gamma\mathcal{O}) = \delta_0 + \delta(\mathfrak{a}_j, \mathfrak{a}_0) - \sum_{i:\begin{cases} e_i \in \mathbf{e}_j \\ e_i < 0 \end{cases}} |e_i| \cdot \deg \mathrm{N}(\mathfrak{p}_i)\,.$$

For any particular B-smooth ideal $\mathfrak{a}'$, it is expected that only a few of the primes in $\mathcal{F}_B$ will be factors of $\mathfrak{a}'$ (supposing that $\mathcal{F}_B$ is large). Hence, we expect the relation vector $\mathbf{e}$ corresponding to $\mathfrak{a}'$ to have only $O(\log n_B) \subseteq \widetilde{O}(1)$ non-zero entries. It follows that the matrix $A_\mathcal{R}$ will also be sparse with expected $\widetilde{O}(n_B)$ non-zero entries.

Note that we will have to generate more than n_B relations. At the very least,

when computing $R_{\mathcal{O}}$ we require $m = n_B + 1$ relations to obtain a full rank matrix $A'_{\mathcal{R}}$, augmented from $A_{\mathcal{R}}$ with the distances in the bottom row. Therefore, we generate the first n_B relations as described above, but for subsequent relations we only need a sequence of principal $\mathcal{O}$-ideals, preferably with small distance. Hence, we will use the baby walk starting from $\mathfrak{a}_0 = \mathcal{O}$. This means that for these new relations we must remember the position in the baby walk to continue the walk for the next relation. The complete relation generation method is presented in Algorithm 4.12.

Algorithm 4.12 (Relation generation). *Generate the a relation for index calculus in the infrastructure using a baby walk.*

Input: A factor base $\mathcal{F}_B = \{\mathfrak{p}_1 = (a_{\mathfrak{p}_1}, b_{\mathfrak{p}_1}), \ldots, \mathfrak{p}_{n_B} = (a_{\mathfrak{p}_{n_B}}, b_{\mathfrak{p}_{n_B}})\}$ and a set $\mathcal{R}_B$ of previously generated relations. If $\#\mathcal{R}_B > n_B$, then we require the $\mathcal{O}$-ideal $\mathfrak{a}$ used to generate the last relation and its distance $\delta(\mathfrak{a})$.

Output: A set of relations $\mathcal{R}_B$ of size one larger than the input.

1: $k \leftarrow \#\mathcal{R}_B + 1$, $j \leftarrow 0$
2: **if** $k \leq n_B$ **then** ▷ Choose $\mathfrak{a}_0$ to get a strictly diagonally dominant matrix
3: $\quad\mathfrak{a}_0 \leftarrow (\overline{\mathfrak{p}_k})^{2g+1}$ ▷ Start with the primitive part of $\mathfrak{p}_k^{-2g-1}$
4: $\quad\delta_0 \leftarrow (2g+1)\deg a_{\mathfrak{p}_k}$ ▷ Distance $\delta\big((\overline{\mathfrak{p}_k})^{2g+1}, \mathfrak{p}_k^{-2g-1}\big)$
5: $\quad$**while** $\mathrm{N}(\mathfrak{a}_0) > g$ **do** ▷ Reduce $\mathfrak{a}_0$
6: $\quad\quad(\mathfrak{a}_0, \delta_0) \leftarrow \rho(\mathfrak{a}_0, \delta_0)$ ▷ Perform a reduction step (Alg. 3.10)
7: **else if** $k = n_B + 1$ **then** ▷ Finished the diagonal dominance, start walk at $\mathcal{O}$
8: $\quad\mathfrak{a}_0 \leftarrow \mathcal{O}$, $\delta_j \leftarrow 0$
9: **else** ▷ Continue the baby walk from the previous ideal in the walk
10: $\quad\mathfrak{a}_0 \leftarrow \mathfrak{a}$, $\delta_j \leftarrow \delta(\mathfrak{a})$
11: $(\mathfrak{a}_{j+1}, \delta_{j+1}) \leftarrow \rho(\mathfrak{a}_j, \delta_j)$ ▷ Next in the baby walk (Alg. 3.10)

$j \leftarrow j + 1$

repeat

  $\mathbf{e} \leftarrow \sigma(\mathcal{F}_B, \mathfrak{a}_j)$ ▷ Test if $\mathfrak{a}_j$ is B-smooth (Alg. 4.10)

  if $\mathbf{e} \neq \mathbf{0}$ **then**

    $\delta \leftarrow \delta_j$

    for each non-zero $e_i \in \mathbf{e}$ **do**

      if $e_i < 0$ **then** ▷ Adjust the distance for inverses

        $\delta \leftarrow \delta + e_i \cdot \deg a_{\mathfrak{p}_i}$ ▷ Subtract the inverse coefficient degree

    if $k \leq n_B$ **then** ▷ If we have not completed diagonal dominance

      $e_k \leftarrow e_k + 2g + 1$ ▷ Correct the k-th entry for $\mathfrak{p}_k^{-2g-1}$

    $\mathcal{R}_B \leftarrow \mathcal{R}_B \cup \{(\mathbf{e}, \delta)\}$ ▷ Store the relation

  else

    $(\mathfrak{a}_{j+1}, \delta_{j+1}) \leftarrow \rho(\mathfrak{a}_j, \delta_j)$ ▷ Next in the baby walk (Alg. 3.10)

    $j \leftarrow j + 1$

until $\mathbf{e} \neq \mathbf{0}$

To estimate the number of reduced B-smooth ideals in $\mathcal{O}$, we use the following result from Enge and Stein [ES02, §5]:

Theorem 4.13 (Enge & Stein, 2002). *If the smoothness bound is chosen such that* $B = \lceil \log_q \mathbf{L}_{q^g}(c) \rceil$ *for some positive real constant* c*, then the number of reduced* $\mathcal{O}$*-ideals that are* B*-smooth is bounded as*

$$n_{\mathcal{O}/B} \geq \frac{q^g}{\mathbf{L}_{q^g}\left(\frac{1}{2c} + o(1)\right)}, \qquad (g \to \infty).$$

Note that the choice of B in Theorem 4.13 matches the analysis from Section 4.3.2.

Since the number of reduced ideals in an ideal class is $R_{\mathcal{O}}$ and there are $h_{\mathcal{O}}$ ideal classes, to get the probability that a reduced ideal in the baby walk corresponds to a B-smooth ideal we divide $n_{\mathcal{O}/B}$ by $h_{\mathcal{O}}R_{\mathcal{O}}$ based on the assumption of Heuristic 4.11.

Now, considering Algorithm 4.12, we expect to repeat the loop testing for a B-smooth ideal $\frac{h_{\mathcal{O}}R_{\mathcal{O}}}{n_{\mathcal{O}/B}}$ times. Theorems 3.2 and 3.7 give an upper bound of $h_{\mathcal{O}}R_{\mathcal{O}} \leq 2q^{g-1}(\deg f - 1)^2$. Since the function field is real quadratic, we have $\deg f \leq 2g+2 \in O(g)$. Recall that we can test for smoothness using Algorithm 4.10 in $O(g^3 \log^2 q + g^2 \log^3 q)$ and compute steps in the baby walk using Algorithm 3.13 in $O(gL(g))$. The number of factors of a B-smooth ideal is in $O(g)$, so adjusting the distance for inverses requires $O(g)$ operations. By Equation 4.3, we have $n_B \in O\left(\mathbf{L}_{q^g}\left(c+\frac{1}{\sqrt{\vartheta}}\right)\right)$ times, where $0 < \vartheta < g/\log q$. From Section 4.3.2, we have the bound $q \in O\left(\mathbf{L}_{q^g}\left(\frac{1}{\sqrt{\vartheta}}\right)\right)$. Then the total heuristic, expected running time of Algorithm 4.12 is

$$\begin{aligned}
&\mathbf{L}_{q^g}\left(\tfrac{1}{2c}+o(1)\right) q^{-g} \cdot 2q^{g-1}(\deg f - 1)^2 \cdot O\Big(g^3 \log^2 q + g^2 \log^3 q\Big) \\
&\in O\left(\mathbf{L}_{q^g}\left(\tfrac{1}{2c}+o(1)\right) \cdot \mathbf{L}_{q^g}\left(\tfrac{-1}{\sqrt{\vartheta}}\right)\left(g^5 \log^2 q + g^4 \log^3 q\right)\right) \\
&\subseteq O\left(\mathbf{L}_{q^g}\left(\tfrac{1}{2c} - \tfrac{1}{\sqrt{\vartheta}} + o(1)\right)\right) \qquad\qquad (g \log q \to \infty)\,.
\end{aligned}$$

We must generate $m \in \Theta(n_B + 1)$ relations. So we must repeat Algorithm 4.10 $O\left(\mathbf{L}_{q^g}\left(c+\frac{1}{\sqrt{\vartheta}}\right)\right)$ times. Then we can express the total heuristic, expected complexity of the relation generation step as

$$\begin{aligned}
&O\left(\mathbf{L}_{q^g}\left(c+\frac{1}{\sqrt{\vartheta}}\right)\right) \cdot O\left(\mathbf{L}_{q^g}\left(\frac{1}{2c} - \frac{1}{\sqrt{\vartheta}} + o(1)\right)\right) \\
&\subseteq O\left(\mathbf{L}_{q^g}\left(c+\frac{1}{2c}+o(1)\right)\right), \qquad (4.7)
\end{aligned}$$

when $g \log q \to \infty$ for $0 < \vartheta \leq g/\log q$.

Comparison with Previous Work

Müller, Stein and Thiel have a runtime of $O\left(\mathbf{L}_{|D|}\left(2c+\frac{1}{4c}\right)\right)$ for relation generation in the infrastructure, where in the odd characterstic case $|D| = q^{\deg f} = q^{2g+2}$ [MST99, §§1.1, 5]. While it is difficult to convert the base of the subexponential function exactly to ours, an indication of how theirs might perform in comparison is shown in the following manipulation:

$$\begin{aligned}
\mathbf{L}_{q^{2g+2}}\left(2c+\tfrac{1}{4c}\right) &= \exp\left(\left(2c+\tfrac{1}{4c}\right)\sqrt{\log q^{2g+2}\log\log q^{2g+2}}\right) \\
&= \exp\left(\left(2c+\tfrac{1}{4c}\right)\sqrt{\tfrac{2g+2}{g}}\sqrt{\log q^{g}\log\log q^{2g+2}}\right) \\
&\geq \mathbf{L}_{q^g}\left(\left(2c+\tfrac{1}{4c}\right)\sqrt{\tfrac{2g+2}{g}}\right). \qquad (4.8)
\end{aligned}$$

If the constant $c \geq 1$, then the function in Equation (4.8) will grow faster with g than ours. We give a comparison of the complete algorithms at the end of Section 4.3.8.

Enge's runtime complexity is $O\left(\mathbf{L}_{q^g}\left(2c+\frac{1}{2c}+\frac{2}{\sqrt{\vartheta}}+o(1)\right)\right)$ for performing relation generation in the Jacobian [Eng02, §5.3]. Enge's method of obtaining candidates to test for smoothness requires computing a linear combination of $O(n_B)$ terms, resulting in another factor of $O\left(\mathbf{L}_{q^g}\left(c+\frac{1}{\sqrt{\vartheta}}\right)\right)$. Enge also obtains an extra factor of $\frac{1}{\sqrt{\vartheta}}$ from the use of the Hasse-Weil bound (Theorem 3.1) as an upper bound for $h_{\mathcal{O}}R_{\mathcal{O}}$, which is not as tight as Artin's result (Theorem 3.2) in real quadratic function fields.

4.3.5 The Lattice of Relations

Let $\mathcal{F}_B = \{\mathfrak{p}_1, \mathfrak{p}_2, \ldots, \mathfrak{p}_{n_B}\}$ be a factor base of prime ideals. Recall from Section 4.3.4 that we generate relations of the form $(\mathbf{e}_j, \delta_j)$ where $\mathbf{e}_j = (e_1, e_2, \ldots, e_{n_B})$ satisfies

$$\prod_{k=1}^{n_B} \mathfrak{p}_k^{e_k} = \alpha\mathcal{O} \qquad \text{for some } \alpha \in F \text{ such that} \qquad \delta_j = \deg \alpha \,. \tag{4.9}$$

Consider the set $\Lambda_B \subset \mathbb{Z}^{n_B}$ consisting of all possible relation vectors $\mathbf{e}_j$ satisfying Equation 4.9 for $\mathcal{F}_B$. We can also consider the augmented set $\Lambda'_B \subset \mathbb{Z}^{n_B+1}$ consisting of all possible relation vectors with the corresponding distances that satisfy Equation 4.9. It is easy to see that Λ_B and Λ'_B are lattices since adding two relations $(e_{i,1}, \ldots, e_{i,n_B}, \delta_i)$ and $(e_{j,1}, \ldots, e_{j,n_B}, \delta_j)$ with $\delta_i = \deg \alpha_i$ and $\delta_j = \deg \alpha_j$ for some $\alpha_i, \alpha_j \in F$ results in another relation corresponding to

$$\prod_{k=1}^{n_B} \mathfrak{p}_k^{e_{i,k}+e_{j,k}} = \alpha'\mathcal{O} \,,$$

where $\alpha' = \alpha_i\alpha_j$ satisfies $\delta_i + \delta_j = \deg \alpha'$.

The following theorem regarding the relation lattices was given in part by Seysen [Sey87, §1] in the imaginary quadratic number field case, and extended to real quadratic number fields by Buchmann [Buc90, §2]. It was generalized to quadratic function fields by Müller, Stein and Thiel [MST99, §2.2]:

Theorem 4.14 (Müller, Stein & Thiel, 1999). *Suppose that the prime ideals in $\mathcal{F}_B$ generate the class group $\mathrm{Cl}(\mathcal{O})$. Then the lattice Λ_B has determinant $h_\mathcal{O}$ and the lattice Λ'_B has determinant $h_\mathcal{O} R_\mathcal{O}$. Furthermore, $\mathbb{Z}^{n_B}/\Lambda_B \cong \mathrm{Cl}(\mathcal{O})$.*

Recall that the prime ideals in $\mathcal{F}_B$ generate the class group $\mathrm{Cl}(\mathcal{O})$ if B is chosen to

satisfy Theorem 4.7. Then Theorem 4.14 tells us how the relation lattices correspond to the ideal class number, the regulator, and the class group. We will use this result as the basis for computing these values in the next few sections.

4.3.6 Computing the Regulator

In this section we describe how one can compute the regulator $R_{\mathcal{O}}$ from the relation matrix. Suppose we have a set $\mathcal{R}_B = \{(\mathbf{e}_1, \delta_1), \ldots, (\mathbf{e}_m, \delta_m)\}$ of $m \geq n_B + 1$ relations and a $n_B \times m$ matrix $A_{\mathcal{R}}$ constructed with column vectors $\mathbf{e}_1^{\text{tr}}, \mathbf{e}_2^{\text{tr}}, \ldots, \mathbf{e}_m^{\text{tr}}$. Consider the $(n_B+1) \times m$ matrix $A'_{\mathcal{R}}$ consisting of the distances $\delta_1, \delta_2, \ldots, \delta_m$ added as a row to the bottom of $A_{\mathcal{R}}$. From Theorem 4.14 we know that if the columns of $A'_{\mathcal{R}}$ generate the lattice Λ'_B, then the determinant of $A'_{\mathcal{R}}$ will be $\Delta = h_{\mathcal{O}} R_{\mathcal{O}}$. It follows that the regulator will be the smallest divisor d of Δ such that $\rho^*(\mathcal{O}, d) = (\mathcal{O}, \delta)$ with $\delta > 0$.

We know from our method of generating relations in Section 4.3.4 that $A'_{\mathcal{R}}$ has a $n_B \times n_B$ submatrix that is strictly diagonally dominant. Therefore, $A'_{\mathcal{R}}$ has rank at least n_B. We assume that relations obtained from the baby walk satisfy the following heuristic:

Heuristic 4.15. *The B-smooth $\mathcal{O}$-ideals obtained from the baby walk result in relations that are randomly sampled from Λ'_B.*

Based on computations with our implementation we claim that Heuristic 4.15 is reasonable. Recall from Theorem 4.14 that the lattice Λ'_B has determinant $h_{\mathcal{O}} R_{\mathcal{O}}$. Under the assumption of Heuristic 4.15, we use the results of Enge [Eng02, §5.4] to claim that, with high probability, $A'_{\mathcal{R}}$ will have full rank and its columns will generate the lattice Λ'_B if we obtain $m = 40 n_B$ relations.

The **Hermite normal form** (HNF) of a $n \times m$ matrix A (with $m \geq n$) is a square $n \times n$ matrix H corresponding to the non-zero columns when the matrix is put in the following special form:

$$\begin{pmatrix} 0 & 0 & \cdots & 0 & h_{1,1} & h_{1,2} & \cdots & h_{1,n} \\ 0 & 0 & \cdots & 0 & 0 & h_{2,2} & \cdots & h_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & h_{n,n} \end{pmatrix},$$

with $0 \leq h_{i,j} < h_{i,i}$ for all $1 \leq i \leq n$ and $i < j \leq n$ [Coh93, p. 66]. The problem with computing the HNF of an integer matrix is that one cannot perform divisions unless they are exact, resulting in "coefficient explosion." Storjohann and Labahn showed that one can reduce the effect of this coefficient explosion and obtain a matrix in HNF that has coefficients about the same size as those in the input matrix.[7] For an $n \times m$ matrix A we use the following notation:

$$\|A\| = \max\left\{ |a_{ij}| \;\middle|\; 1 \leq i \leq n,\ 1 \leq j \leq m \right\},$$

where a_{ij} denotes the (i,j)-th entry of A. Then we can compute the HNF in $\widetilde{O}\big(n^{\theta-1}m \cdot M(n \log\|A\|)\big)$ bit operations, where $M(t)$ is the number of bit operations required to multiply two $\lceil t \rceil$-bit integers, and θ is the exponent for matrix multiplication in $\mathbb{Z}$ [SL96, §1]. The best known algorithm for matrix multiplication has $\theta = 2.38$ due to Coppersmith and Winograd [CW90]. Then, because the augmented relation matrix

[7] Hafner and McCurley's method for computing the HNF of an $n \times n$ matrix results in entries that are a factor of n larger than the input matrix [HM91].

$A'_{\mathcal{R}}$ has dimension $(n_B + 1) \times m$, its HNF can be computed in

$$\widetilde{O}\left(\mathsf{L}_{q^g}\left(2.38c + \frac{2.38}{\sqrt{\vartheta}}\right) \cdot M(n_B \log\|A'_{\mathcal{R}}\|)\right) \subseteq O\left(\mathsf{L}_{q^g}\left(2.38c + \frac{2.38}{\sqrt{\vartheta}} + o(1)\right)\right).$$

After we compute the determinant Δ of $A'_{\mathcal{R}}$, we have that $\Delta = kR_{\mathcal{O}}$ for a small multiple k (with high probability $k = h_{\mathcal{O}}$, but it need not be). Then we can find the regulator by factoring the integer Δ and finding the largest divisor that is the distance of a unit. This technique was also used by Stein and Williams [SW98, §5.1]. Müller, Stein and Thiel suggested transforming the HNF of $A'_{\mathcal{R}}$ into Smith normal form to compute the class number $h_{\mathcal{O}}$ (*cf.* Section 4.3.7) [MST99, §2.4]. This overhead is avoided by our technique, presented in Algorithm 4.16.

Algorithm 4.16 (Finding the regulator from a multiple). *Compute the regulator from some multiple by factoring and using the infrastructure to find the largest divisor that is the distance of a unit.*

Input: The factor base $\mathcal{F}_B = \{\mathfrak{p}_1, \mathfrak{p}_2, \ldots, \mathfrak{p}_{n_B}\}$, and an integer $\Delta = kR_{\mathcal{O}}$ for some unknown $k \in \mathbb{Z}$.

Output: The regulator $R_{\mathcal{O}}$.

1: $(\mathfrak{a}, \delta) \leftarrow \rho^*(\mathcal{O}, \Delta)$ ▷ Compute the closest ideal to the distance Δ (Alg. 3.26)
2: **if** $\mathfrak{a} \neq \mathcal{O} \vee \delta = 0$ **then** ▷ Ensure we start with a multiple of R
3: **return** failure
4: $p_1^{n_1} p_2^{n_2} \cdots p_\ell^{n_\ell} \leftarrow \mathsf{Factor}(\Delta)$ ▷ Using an efficient factoring algorithm
5: **for** i **from** 1 **to** ℓ **do**
6: **for** j **from** 1 **to** n_i **do**
7: $d \leftarrow \Delta / p_i$

8: $(\mathfrak{a}, \delta) \leftarrow \rho^*(\mathcal{O}, d)$ ▷ The closest ideal to the distance d (Alg. 3.26)
9: **if** $\mathfrak{a} = \mathcal{O} \wedge \delta \neq 0$ **then**
10: $\Delta \leftarrow d$ ▷ d is still a multiple of R
11: **else**
12: **break** ▷ Exit the inner for-loop
13: $R_{\mathcal{O}} \leftarrow \Delta$

It is important in Algorithm 4.16 to use an efficient factoring algorithm, such as the number field sieve [LLMP90], since the determinant Δ is likely to be large. NFS runs asymptotically in expected $O\big(\mathsf{L}_{\Delta}\left(1/3, (64/9)^{1/3} + o(1)\right)\big)$ bit operations [BLP93]. The number of prime factors of the determinant Δ is $\ell \in O(\log \Delta)$. Similarly, we bound the exponents $n_i \in O(\log \Delta)$. Then we can compute the closest ideal to a given distance d in $\widetilde{O}(dg)$ field operations. The runtime of Algorithm 4.16 is obviously dominated by the factorization. However, since Δ will be a small multiple of $R_{\mathcal{O}}$ and the factorization is subexponential to the 1/3, the asymptotic runtime for computing the determinant and finding the regulator is dominated by the HNF calculation.

Table 4.17 summarizes the complexity of each step involved in computing the regulator.

Table 4.17. *Asymptotic complexity of computing the regulator*

1. Factor base generation	$O\left(\mathsf{L}_{q^g}\left(c + \frac{1}{\sqrt{\vartheta}} + o(1)\right)\right)$
2. Relation generation (heuristic, expected)	$O\left(\mathsf{L}_{q^g}\left(c + \frac{1}{2c} + o(1)\right)\right)$
3. Computing the regulator	$O\left(\mathsf{L}_{q^g}\left(2.38c + \frac{2.38}{\sqrt{\vartheta}} + o(1)\right)\right)$

Total (heuristic, expected)	$O\left(\mathbf{L}_{q^g}\left(\max\left\{2.38c + \frac{2.38}{\sqrt{\vartheta}},\ c + \frac{1}{2c}\right\} + o(1)\right)\right)$

Based on the results in Table 4.17, we can choose the constant $c \in \mathbb{R}_{>0}$ to minimize the overall complexity. The function $f(c) = c + \frac{1}{2c}$ has a unique minimum of $\sqrt{2}$ at $c = \frac{1}{\sqrt{2}}$. Equating $f(c)$ with the function $g(c) = 2.38c + \frac{2.38}{\sqrt{\vartheta}}$, we obtain a constant of

$$c = \frac{\sqrt{6900\vartheta + 14161} - 119}{138\sqrt{\vartheta}}.$$

For $\vartheta = 1$, we take the minimum value for c and obtain an overall heuristic, expected runtime in $O(\mathbf{L}_{q^g}(2.83 + o(1)))$ to compute the regulator using index calculus in the infrastructure.

4.3.7 Determining the Class Number and Group Structure

Suppose we have the $(n_B + 1) \times m$ augmented relation matrix $A'_{\mathcal{R}}$ described in the previous section. If we compute the Hermite normal form of $A'_{\mathcal{R}}$ as in the last section, we know that the determinant Δ of the HNF will likely be $h_{\mathcal{O}}R_{\mathcal{O}}$, but could be a small multiple of $h_{\mathcal{O}}R_{\mathcal{O}}$. The determinant will be exactly $h_{\mathcal{O}}R_{\mathcal{O}}$ if the columns of $A'_{\mathcal{R}}$ generate the full relation lattice Λ'_B. To determine with certainty whether we generate Λ'_B, we must compute an estimate of the product $h_{\mathcal{O}}R_{\mathcal{O}}$ that is precise enough that we can detect whether Δ is exactly $h_{\mathcal{O}}R_{\mathcal{O}}$ or a multiple.

Recall from Theorem 3.7 that $h_F = h_{\mathcal{O}}R_{\mathcal{O}}$, where h_F is the divisor class number of F. To ensure the determinant Δ is not a multiple, we must obtain a value h_F^* such that $h_F^* < h_F < 2h_F^*$. Unfortunately, the Hasse-Weil bound (Theorem 3.1)

is not sufficient. A more precise bound on h_F can be obtained using the analytic class number formula. For any monic irredicible polynomial $p \in K[x]$, we define a **character** $\chi(p)$ based on the splitting behaviour of the prime ideal $\mathfrak{p} = pK[x]$ in $\mathcal{O}$ (*cf.* Section 4.3.2):

$$\chi(p) = \begin{cases} 0 & \text{if } \mathfrak{p} \text{ is ramified in } \mathcal{O} \\ +1 & \text{if } \mathfrak{p} \text{ splits in } \mathcal{O} \\ -1 & \text{if } \mathfrak{p} \text{ is inert in } \mathcal{O}\,. \end{cases}$$

Let $L_\chi(s)$ be the L-**polynomial** associated with the character χ in F (*cf.* Stichtenoth [Sti93, pp. 165–166]). The following theorem relates the divisor class number to the L-polynomial.

Theorem 4.18 (Analytic class number formula). *For any algebraic function field F over a finite field $\mathbb{F}_q$, the divisor class number $h_F = L_\chi(1)$.*

The functional equation of the L-polynomial states that $L_\chi(1) = q^g \cdot L_\chi(1/q)$ [Sti93, p. 166]. We can write this functional equation as an **Euler product** to obtain

$$h_F = L_\chi(1) = q^g \cdot L_\chi(1/q) = \frac{q^{g+1}}{q-1} \prod_{p \in K[x]} \frac{1}{1 - \chi(p) q^{-\deg p}}\,, \tag{4.10}$$

where we are again assuming the $p \in K[x]$ are monic irreducible polynomials and F is real quadratic [ST02a, §4.1]. An approximation $\widetilde{h_F}(\lambda)$ for Equation (4.10) can be obtained by truncating the Euler product by bounding $\deg p$ by a parameter $\lambda \in \mathbb{N}$.

That is, we compute $\widetilde{h_F}(\lambda) \in \mathbb{R}$ as

$$\widetilde{h_F}(\lambda) = \frac{q^{g+1}}{q-1} \prod_{\substack{p \in K[x] \\ \deg p \leq \lambda}} \frac{1}{1 - \chi(p) q^{-\deg p}} ,$$

We will set $h_F^* = \mu \cdot \widetilde{h_F}(\lambda)$, for some value μ to be determined such that $\frac{1}{2} < \mu < 1$. Note that $h_F = e^{D(\lambda)} \cdot \widetilde{h_F}(\lambda)$ for some $D(\lambda) \in \mathbb{R}$, where the value $e^{D(\lambda)}$ measures the error of our class number approximation. Following Düllmann [Dül91, pp. 38–39] and Abel [Abe94, pp. 53–57] in quadratic number fields, we can get bounds on the size of $D(\lambda)$ required to ensure $h_F^* < h_F < 2h_F^*$ as follows:

$$\begin{array}{rcccl}
h_F^* & < & h_F & < & 2h_F^* \\
\mu \cdot \widetilde{h_F}(\lambda) & < & h_F & < & 2\mu \cdot \widetilde{h_F}(\lambda) \\
\mu & < & \dfrac{h_F}{\widetilde{h_F}(\lambda)} & < & 2\mu \\
\mu & < & e^{D(\lambda)} & < & 2\mu \\
\log \mu & < & D(\lambda) & < & \log(2\mu) .
\end{array}$$

Then we have $|D(\lambda)| < \min\{-\log \mu,\ \log(2\mu)\}$, which has a minimum value of $\frac{1}{2}\log 2$ when $\mu = \frac{\sqrt{2}}{2}$. Therefore, to satisfy our bounds we must choose λ sufficiently large to satisfy $|D(\lambda)| < \frac{1}{2}\log 2$.

To determine an appropriate parameter λ, we use the following result from Stein and Teske [ST02a, §4.3]:

Theorem 4.19 (Stein & Teske, 2002). *For a quadratic function field F, we can approximate the divisor class number h_F with $\widetilde{h_F}(\lambda)$ such that $h_F = e^{D(\lambda)} \cdot \widetilde{h_F}(\lambda)$ and*

$$|D(\lambda)| < \frac{(2g + (\lambda \bmod 2))q^{-\frac{\lambda}{2}}}{(\lambda+1)(\sqrt{q}-1)} + \frac{(2g+2)q^{-\frac{(\lambda-1)}{2}}}{(\lambda+2)(\sqrt{q}-1)^3} .$$

We have that $\widetilde{h_F}(\lambda)$ can be computed in $O(q^\lambda)$ field operations [ST02a, §5.1]. Therefore, to remain subexponential in $g \log q$, we require $\lambda = \log_q(X)$ for some X subexponential in $g \log q$. We derive from Theorem 4.19 the following rough upper bound:

$$|D(\lambda)| < q^{-\frac{\lambda}{2}} X, \qquad \text{for} \quad X = \frac{2g+1}{\sqrt{q}-1} + \frac{(2g+2)\sqrt{q}}{(\sqrt{q}-1)^3} .$$

Then to compute $\widetilde{h_F}(\lambda)$ with sufficient accuracy for $h_F \in \left(h_F^*, 2h_F^*\right)$ we require

$$\begin{aligned} \frac{1}{2}\log 2 &> q^{-\frac{\lambda}{2}} X \\ q^{\frac{\lambda}{2}} &> 2X \log^{-1} 2 \\ \lambda &> 2\log_q\left(2X \log^{-1} 2\right) . \end{aligned}$$

Since $X \in O(g\sqrt{q}) \subseteq \mathbf{L}_{q^g}\left(\frac{1}{2\sqrt{\vartheta}} + o(1)\right)$, we can compute $\widetilde{h_F}(\lambda)$ and $h_F^* = \frac{\sqrt{2}}{2}\widetilde{h_F}(\lambda)$ in subexponential time as $g \to \infty$.

Now, after computing the determinant Δ of the HNF of $A'_{\mathcal{R}}$, we verify that $h_F^* < \Delta < 2h_F^*$. If not, then we only generate a sublattice $\Lambda'_{\mathcal{R}} \subset \Lambda'_B$ and we must add more relations to $\mathcal{R}_B$ and the matrix $A'_{\mathcal{R}}$ using Algorithm 4.12 until $h_F^* < \Delta < 2h_F^*$ is satisfied, at which point $\Lambda'_{\mathcal{R}} = \Lambda'_B$. In order to ensure that we obtain relations in $\Lambda'_B \setminus \Lambda'_{\mathcal{R}}$, we assume Heuristic 4.15. Then we can use the results of Müller, Stein and Thiel [MST99, §4] to claim that we can obtain new relations such that $A'_{\mathcal{R}}$ will

generate the full lattice Λ'_B. The total runtime complexity to ensure our relations generate the lattice Λ'_B is

$$O\left(\mathbf{L}_{q^g}\left(\max\left\{2.38c+\frac{2.38}{\sqrt{\vartheta}},\ c+\frac{1}{2c},\ \frac{1}{2\sqrt{\vartheta}}\right\}+o(1)\right)\right).$$

Once we have $\Delta = h_{\mathcal{O}}R_{\mathcal{O}}$ by verifying that $h^*_F < \Delta < 2h^*_F$, we can separate the class number $h_{\mathcal{O}}$ from Δ by computing the regulator $R_{\mathcal{O}}$ using Algorithm 4.16. However, here we give a different method for computing the class number that also reveals the class group structure.

The **Smith normal form** of an $n \times m$ matrix A (with $m \geq n$) is the unique $n \times m$ matrix $S = UAV$ with invertible matricies $U \in \mathbb{Z}^{n\times n}$ and $V \in \mathbb{Z}^{m\times m}$, where S is in the following form with non-zero diagonal entries $s_1, \ldots, s_r$ for $r = \text{rank}(A)$:

$$S = \begin{pmatrix} s_1 & 0 & & \cdots & \cdots & & 0 \\ 0 & s_2 & & & & & \\ & & \ddots & \ddots & & & \vdots \\ \vdots & & \ddots & s_r & \ddots & & \vdots \\ \vdots & & & \ddots & 0 & & \\ & & & & & \ddots & 0 \\ 0 & & \cdots & \cdots & & 0 & 0 \end{pmatrix}.$$

The diagonal entries $s_1, \ldots, s_r$ are called the **invariant factors** of A and they satisfy the property that $s_i | s_{i+1}$ for $1 \leq i \leq r-1$. Giesbrecht [Gie01] gave an algorithm for

computing the SNF of a sparse $n \times m$ matrix A with $m > n$ in

$$\widetilde{O}(m^2 n \log\|A\| + m^3 \log^2\|A\|) .$$

Hafner and McCurley [HM89, §2] showed that one can obtain the class number and class group structure from the Smith normal form of $A_{\mathcal{R}}$ in the context of imaginary quadratic number fields. This method was proposed for the real quadratic function field case by Müller, Stein and Thiel [MST99, §2.4]. If we have the HNF H' of $A'_{\mathcal{R}}$ such that $A'_{\mathcal{R}}$ generates Λ'_B, then we remove distances in the bottom row from H' to get H and compute the SNF of H. If B is chosen such that the prime ideals in $\mathcal{F}_B$ generate the class group $\mathrm{Cl}(\mathcal{O})$, then the class number $h_{\mathcal{O}}$ will be the product of the invariant factors of the SNF of H. Moreover, the invariant factors $s_1, \ldots, s_{n_B}$ that are greater than one will correspond to the structure of the class group:

$$\mathrm{Cl}(\mathcal{O}) \cong \bigoplus_{\substack{i=1 \\ s_i>1}}^{n_B} \mathbb{Z}/s_i\mathbb{Z} .$$

Then using Giesbrecht's algorithm and the fact that H has dimension $n_B \times n_B$, we can compute the class number and class group structure in

$$\widetilde{O}\left(\mathsf{L}_{q^g}\left(3c + \frac{3}{\sqrt{\vartheta}}\right)\log^2\|H\|\right) \subseteq O\left(\mathsf{L}_{q^g}\left(3c + \frac{3}{\sqrt{\vartheta}} + o(1)\right)\right) .$$

Table 4.20 summarizes the complexity of each step involved in computing the class number and group structure.

Table 4.20. *Asymptotic complexity of computing the class number and class group structure*

1. Factor base generation	$O\left(\mathsf{L}_{q^g}\left(c+\frac{1}{\sqrt{\vartheta}}+o(1)\right)\right)$
2. Relation generation (heuristic, expected)	$O\left(\mathsf{L}_{q^g}\left(c+\frac{1}{2c}+o(1)\right)\right)$
3. Ensuring $A'_{\mathcal{R}}$ generates Λ'_R (heuristic, expected)	$O\left(\mathsf{L}_{q^g}\left(\max\left\{2.38c+\frac{2.38}{\sqrt{\vartheta}},\ c+\frac{1}{2c},\ \frac{1}{2\sqrt{\vartheta}}\right\}+o(1)\right)\right)$
4. Computing the class number and group structure	$O\left(\mathsf{L}_{q^g}\left(3c+\frac{3}{\sqrt{\vartheta}}+o(1)\right)\right)$
Total (heuristic, expected)	$O\left(\mathsf{L}_{q^g}\left(\max\left\{3c+\frac{3}{\sqrt{\vartheta}},\ c+\frac{1}{2c}\right\}+o(1)\right)\right)$

Using the results in Table 4.20, we choose the constant $c \in \mathbb{R}_{>0}$ to minimize the overall complexity. As in the regulator case, the function $f(c) = c + \frac{1}{2c}$ has a unique minimum at $c = \frac{1}{\sqrt{2}}$. We equate $f(c) = c + \frac{1}{2c}$ with the function $g(c) = 3c + \frac{3}{\sqrt{\vartheta}}$ to obtain a constant of

$$c = \frac{\sqrt{4\vartheta + 9} - 3}{4\sqrt{\vartheta}}.$$

For $\vartheta = 1$, the minimum value for c gives an overall heuristic, expected runtime in $O(\mathsf{L}_{q^g}(3.45 + o(1)))$ to compute the class number and group structure using index calculus in the infrastructure.

4.3.8 Computing Discrete Logarithms

The linear algebra step for the infrastructure DLP is formulated slightly different than in the general algorithm described in §4.2. We assume that we already know the regulator $R_{\mathcal{O}}$, and we will solve the linear system modulo $R_{\mathcal{O}}$. We will also assume that $R_{\mathcal{O}}$ is prime as would likely be desired in cryptosystems. If the regulator $R_{\mathcal{O}}$ is composite, we would have to solve the linear system modulo each prime power factor of $R_{\mathcal{O}}$ and combine the results using the Chinese remainder theorem. Possible issues with this process are discussed by Enge and Gaudry [EG02, §4].

Recall the infrastructure DLP from Problem 4.6: we are given a reduced principal $\mathcal{O}$-ideal $\mathfrak{b}$ and we wish to compute the distance $\delta(\mathfrak{b})$. We start by finding a B-smooth ideal $\mathfrak{b}'$ equivalent to $\mathfrak{b}$ and puts its factorization in a vector $\mathbf{b}$. Then, with the relation matrix $A_{\mathcal{R}}$, we solve for the vector $\mathbf{x}$ in $A_{\mathcal{R}}\mathbf{x} = \mathbf{b} \pmod{R_{\mathcal{O}}}$. Unlike in the general index calculus description in Section 4.2, the solution vector $\mathbf{x}$ does not give discrete logarithms of the primes in $\mathcal{F}_B$, but rather $\mathbf{x}$ will describe how the ideals that factor according to the columns of $A_{\mathcal{R}}$ relate to the ideal $\mathfrak{b}'$ corresponding to $\mathbf{b}$. Since we have the infrastructure DLP solutions for our relations, *i.e.* the distances stored with the relation vectors in $\mathcal{R}$, we use $\mathbf{x}$ to compute the distance $\delta(\mathfrak{b}')$. From there, the distance $\delta(\mathfrak{b}', \mathfrak{b})$ gives us the desired solution $\delta(\mathfrak{b})$.

We now describe the steps in more detail. We find a B-smooth ideal equivalent to $\mathfrak{b}_0 = \mathfrak{b}$ by performing baby steps $\mathfrak{b}_j = \rho(\mathfrak{b}_{j-1})$ for $j = 1, \ldots$ until $\mathfrak{b}_j$ is B-smooth. Using the smoothness test given in Algorithm 4.10 we obtain a vector $\mathbf{b} = (b_1, \ldots, b_{n_B})$ such that $\prod_{i=1}^{n_B} \mathfrak{p}_i^{b_i} = \mathfrak{b}_j$. To bound the size of j, we once again assume Heuristic 4.11, giving $j \leq \frac{h_{\mathcal{O}} R_{\mathcal{O}}}{n_{\mathcal{O}/B}}$ with high probability. Therefore, as in Section 4.3.4, the heuristic,

expected number of ideals we have to test before finding one that is B-smooth is

$$j \in O\left(\mathsf{L}_{q^g}\left(\frac{1}{2c} - \frac{1}{\sqrt{\vartheta}} + o(1)\right)\right).$$

Suppose we have the $n_B \times m$ relation matrix $A_{\mathcal{R}}$ corresponding to a set of relations $\mathcal{R}_B$ of size $m \geq n_B$. In our linear algebra step to compute the infrastructure DLP, we solve for $\mathbf{x}$ in the following system:

$$A_{\mathcal{R}}\mathbf{x} = \mathbf{b} \pmod{R_{\mathcal{O}}}. \tag{4.11}$$

We can solve Equation (4.11) when the relation matrix is sparse using Wiedemann's algorithm [Wie86]. A variant due to Kaltofen and Saunders requires $O(n_B)$ multiplications of $A_{\mathcal{R}}$ by vectors and $O(n_B^2)$ operations in the field $\mathbb{Z}/R_{\mathcal{O}}\mathbb{Z}$ [KS91, §2]. If the number of non-zero entries in $A_{\mathcal{R}}$ is $\omega \in \widetilde{O}(m)$, then we can solve the system in Equation (4.11) in

$$\widetilde{O}(n_B(m + n_B)) \subseteq O\left(\mathsf{L}_{q^g}\left(2c + \frac{2}{\sqrt{\vartheta}} + o(1)\right)\right).$$

We form a vector $\mathbf{d} = (\delta_1, \delta_2, \ldots, \delta_m)$ from the distances stored during relation generation and compute the standard dot-product

$$\delta = \mathbf{d} \cdot \mathbf{x} = \sum_{i=1}^{m} \delta_i x_i \pmod{R_{\mathcal{O}}}.$$

This value δ is equal to the distance $\delta(\mathfrak{b}_j)$. Then $\delta(\mathfrak{b}) = \delta - \delta(\mathfrak{b}_j, \mathfrak{b})$, where $\delta(\mathfrak{b}_j, \mathfrak{b})$ is computed from the baby steps used to obtain $\mathfrak{b}_j$.

This method to compute the infrastructure DLP is a Monte Carlo algorithm. We

can verify the distance $\delta(\mathfrak{b})$ by computing the ideal closest to $\delta(\mathfrak{b})$ from $\mathcal{O}$ using Algorithm 3.14 and verifying the result is $\mathfrak{b}$. If this check is not satisfied, we must add new relations to $A_{\mathcal{R}}$ until $\mathbf{b}$ is in the column-span of $A_{\mathcal{R}}$.

We summarize the complexity of each step involved in solving an instance of the infrastructure DLP in Table 4.21.

Table 4.21. *Asymptotic expected complexity of solving the infrastructure DLP*

1. Factor base generation	$O\left(\mathbf{L}_{q^g}\left(c+\frac{1}{\sqrt{\vartheta}}+o(1)\right)\right)$
2. Relation generation (heuristic, expected)	$O\left(\mathbf{L}_{q^g}\left(c+\frac{1}{2c}+o(1)\right)\right)$
3. Finding a smooth ideal equivalent to $\mathfrak{b}$ (heuristic, expected)	$O\left(\mathbf{L}_{q^g}\left(\frac{1}{2c}-\frac{1}{\sqrt{\vartheta}}+o(1)\right)\right)$
4. Solving the linear system and computing the infrastructure DLP	$O\left(\mathbf{L}_{q^g}\left(2c+\frac{2}{\sqrt{\vartheta}}+o(1)\right)\right)$
Total (heuristic, expected)	$O\left(\mathbf{L}_{q^g}\left(\max\left\{2c+\frac{2}{\sqrt{\vartheta}},\ c+\frac{1}{2c}\right\}+o(1)\right)\right)$

Similar to Section 4.3.6, we choose the constant $c \in \mathbb{R}_{>0}$ based on the results in Table 4.17 to minimize the overall complexity. Again, the function $f(c) = c + \frac{1}{2c}$ has a unique minimum at $c = \frac{1}{\sqrt{2}}$. Equating $f(c)$ with the function $g(c) = 2c + \frac{2}{\sqrt{\vartheta}}$, we obtain a constant of

$$c = \frac{\sqrt{2}\sqrt{\vartheta+2}-2}{2\sqrt{\vartheta}}.$$

For $\vartheta = 1$, with the minimum value for c we obtain an overall heuristic, expected

runtime in $O(\mathbf{L}_{q^g}(2.45 + o(1)))$ to solve an instance of the infrastructure discrete logarithm problem using index calculus.

Comparision with Previous Work

Müller, Stein and Thiel give the asymptotic with a different subexponential base:

$$O\left(\mathbf{L}_{q^{2g+2}}\left(\max\left\{2c + \frac{1}{4c},\ 5c\right\} + o(1)\right)\right).$$

Using this base they were able to calculate a runtime of $O(\mathbf{L}_{q^{2g+2}}(1.44 + o(1)))$ with the constant $c = \frac{1}{2\sqrt{3}}$ [MST99, §5]. Figure 4.22 contains a plot of our runtime $\mathbf{L}_{q^g}(2.45)$ with MST's $\mathbf{L}_{q^{2g+2}}(1.44)$, showing that ours grows slower with the genus.

Enge's method for computing the Jacobian DLP has an asymptotic of

$$O\left(\mathbf{L}_{q^g}\left(\max\left\{5c + \frac{5}{\sqrt{\vartheta}},\ 2c + \frac{1}{2c} + \frac{2}{\sqrt{\vartheta}}\right\} + o(1)\right)\right).$$

The constant in this case is minimized between $c = \frac{1}{2}$ and

$$c = \frac{1}{\sqrt{6}}\left(\sqrt{1 + \frac{3}{2\vartheta}} - \sqrt{\frac{3}{2\vartheta}}\right).$$

With $\vartheta = 1$, Enge's runtime is calculated as $O(\mathbf{L}_{q^g}(5.73 + o(1)))$ [Eng02, §5.5]. We also plotted $\log \mathbf{L}_{q^g}(5.73)$ in our comparison in Figure 4.22.

Note that both Enge and Müller, Stein and Thiel are able to obtain rigourous (non-heuristic) runtimes based on their methods for relation generation. However, our analysis corresponds to an algorithm that is also efficient in implementation.

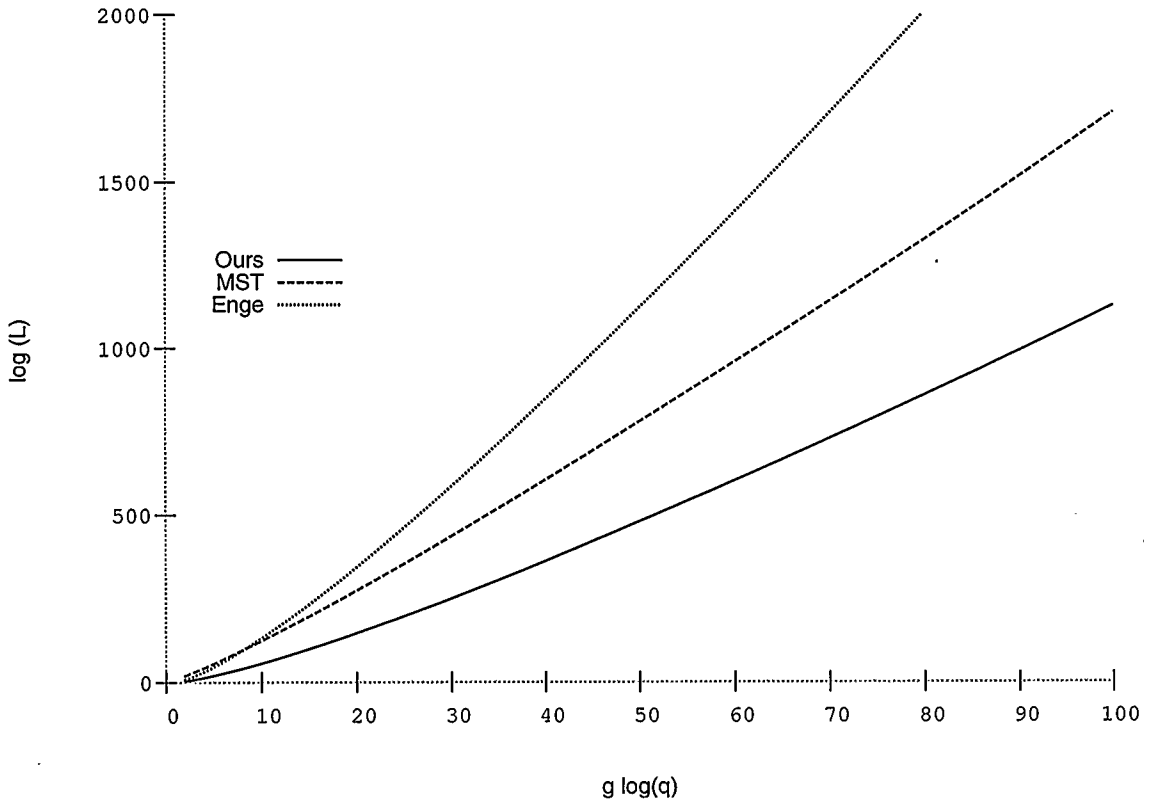

Figure 4.22. *A plot of the logarithm of the subexponential runtimes for computing discrete logarithms when varying* $g \log q$ *with* $\vartheta = \lfloor g/\log q \rfloor = 1$. *Our algorithm and the algorithm given by Müller, Stein and Thiel are both in the infrastructure with asymptotic runtimes of* $\mathbf{L}_{q^g}(2.45)$ *and* $\mathbf{L}_{q^{2g+2}}(1.44)$, *respectively. Enge's result is in the Jacobian with an asymptotic runtime of* $\mathbf{L}_{q^g}(5.73)$.

Chapter 5

Implementing Index Calculus in the Infrastructure

To our knowledge, we are the first to implement an index calculus algorithm in the infrastructure of a real quadratic function field. This is important as it serves as a proof-of-concept to the theoretical discussion and gives concrete timings that are faster than those previously published for various computational problems in the infrastructure. This is especially relevant for estimating the security of cryptosystems based on the infrastructure discrete logarithm problem.

In an attempt to get the most efficient runtimes possible, we have implemented variations on the algorithm that was analyzed asymptotically in the last chapter. These include self-initialized sieving for relation generation, the use of large primes to include partial relations, and changes to the linear algebra step. Each of these methods potentially improve the runtime of the algorithm in practice, but are difficult to analyze to obtain asymptotic runtimes. Therefore, we present the theory behind these variations in Section 5.1, reflecting the goal of this chapter to describe a practical

implementation of index calculus in the infrastructure.

We briefly discuss the specifics of our implementation in Section 5.2. Section 5.3 outlines how we chose our parameters for generating the numerical results presented in Section 5.4.

5.1 Practical Improvements

Techniques introduced in the context of integer factorization have been adapted to many other problems where index calculus is applied. In this section we present some of these variations that we have implemented in the infrastructure setting, as well as a few new techniques that apply directly to our setting. The first technique we introduce is sieving for relation generation. In Section 5.1.2 we extend our discussion of sieving to multiple polynomials and self initialization techniques. We briefly mention low-degree sieving in Section 5.1.3 as another practical improvement. Following, in Section 5.1.4, we present another variation to relation generation that allows the use of large primes and introduce partial relations. Finally, we discuss techniques to improve the performance of the linear algebra step in Section 5.1.5.

5.1.1 Relation Generation via Sieving

Sieve methods for relation generation have been used effectively in factoring algorithms since the late 1970s (*cf.* [Pom96]). Jacobson implemented a self-initialized sieve in quadratic number fields [Jac99]. Flassenberg and Paulus described how to adapt sieving to the Jacobian of a hyperelliptic curve over odd characteristic fields [FP99]. This was extended to the characteristic two case in imaginary hyperelliptic

curves in the recent work by Velichka [Vel08]. Velichka compared discrete logarithm computations using implementations of sieving and random walks and obtained results that showed that sieving is more efficient in the relation generation phase [Vel08, pp. 111–120]. For our purposes, we used Velichka's sieving implementation with minor tweaks to adapt it to the infrastructure of a real quadratic function field. We provide a brief explanation of the implemented sieving technique here, generalized to both even and odd field characteristics. For more details, see Velichka [Vel08, pp. 49–56, 76–88].

Let $\mathcal{O}$ be the quadratic order of a real quadratic function field F defined by $y^2 + h(x)y = f(x)$ over a finite field $K = \mathbb{F}_q$. Consider a primitive $\mathcal{O}$-ideal $\mathfrak{a} = (a, b)$ in standard representation. For any element $\alpha \in \mathfrak{a}$ we have $\alpha = au + (b + y)v$ for some $u, v \in K[x]$. Then we can expand the norm of α using the conjugate under the hyperelliptic involution ι and group it as follows:

$$\begin{aligned}
\mathrm{N}(\alpha) &= \alpha\overline{\alpha} \\
&= (au + (b+y)v) \cdot \iota(au + (b+y)v) \\
&= (au + (b+y)v)(au + (b-h-y)v) \\
&= a^2u^2 + a(2b-h)uv + (b^2 - bh - hy - y^2)v^2 \\
&= a\left(au^2 + (2b-h)uv + \frac{(b^2 - bh - f)}{a}v^2\right).
\end{aligned}$$

Since $\mathrm{N}(\mathfrak{a}) = a$ and the ideal norm is multiplicative, there must exist an $\mathcal{O}$-ideal $\mathfrak{b}$ with $\mathrm{N}(\mathfrak{b}) = au^2 + (2b-h)uv + cv^2$ and $c = (b^2 - bh - f)/a$ such that $\alpha\mathcal{O} = \mathfrak{a}\mathfrak{b}$ [Vel08, p. 50]. Let $g_0(u, v) = \mathrm{N}(\mathfrak{b})$ for $u, v \in K[x]$. We will only consider one-dimensional

sieving,[1] so we fix $v = 1$ and call $g_0(u) = au^2 + (2b - h)u + c$ the **sieving polynomial.**

Sieving will find values of $u \in K[x]$ such that the ideal $\mathfrak{b}$ with $\mathrm{N}(\mathfrak{b}) = g_0(u)$ will correspond to ideals that are likely to be B-smooth. We use a fixed-sized array indexed by polynomials in $K[x]$. For each prime ideal $\mathfrak{p} = (a_\mathfrak{p}, b_\mathfrak{p}) \in \mathcal{F}_B$, the sieving process jumps through the array, increasing the entry at index $j \in K[x]$ if $a_\mathfrak{p} | g_0(j)$. After this process is completed, for each index $j \in K[x]$ with an array entry larger than some tolerance value, we perform a smoothness test on the primitive ideal with norm $g_0(j)$. Since the array entry for j is large, we already know that many of the norms of the prime ideals in the factor base divide $g_0(j)$. Thus, it is highly probable that we will get a B-smooth ideal from $g_0(j)$.

We require a way to represent polynomials in $K[x]$ as integers to index an array. Following Flassenberg and Paulus [FP99, §4.2], let a polynomial $k \in K[x]$ with $n = \deg k$ be denoted by $k = k_n x^n + \cdots + k_1 x + k_0$ and define the following map:

$$\nu : \begin{cases} K[x] & \to & \mathbb{N}_0 \\ k(x) & \mapsto & \sum_{i=0}^{\deg k} \nu_0(k_i) q^i \end{cases},$$

where the map $\nu_0 : K \to \mathbb{N}_0$ sends a field element to an integer between 0 and $q - 1$. This is a one-to-one map, *i.e.* each unique polynomial $k \in K[x]$ has a unique value $\nu(k) \in \mathbb{N}_0$. Then we can compute polynomials $k_0, k_1, \ldots$ to iterate through an array indexed by ν using Algorithm 5.1, denoted by λ.

[1] Flassenberg and Paulus [FP99] described two dimensional sieving, but Velichka argued that only using one dimension is acceptable in practice [Vel08, p. 52].

Algorithm 5.1 (Next array index λ). *Iterates through the polynomials $k \in K[x]$ to compute the next array index according to the map ν.*

Input: The previous polynomial k (initially 0).

Output: A new value k' such that $\nu(k') = \nu(k) + 1$.

1: $e \leftarrow 0$, $k' \leftarrow k$
2: **repeat**
3: $\quad k' \leftarrow k' + x^e$
4: $\quad$ **if** $k' < k$ **then**
5: $\quad\quad e \leftarrow e + 1$
6: **until** $k' > k$

We choose a **sieving interval** M and initialize a **sieve array** D as $D[\nu(k)] = 0$ for $0 \leq \deg(k) \leq M$. Let $\mathcal{F}_B = \{\mathfrak{p}_1, \mathfrak{p}_2, \ldots, \mathfrak{p}_{n_B}\}$ denote the set of prime ideals in the factor base. For each $\mathfrak{p} = (a_\mathfrak{p}, b_\mathfrak{p}) \in \mathcal{F}_B$, let $\mathcal{S}_{g_0,\mathfrak{p}}$ denote the set of roots of $g_0(u)$ modulo $\mathrm{N}(\mathfrak{p}) = a_\mathfrak{p}$. We can find these roots efficiently using the "self-initialization" technique described later. For each $\mathfrak{p} \in \mathcal{F}_B$ and each root $r \in \mathcal{S}_{g_0,\mathfrak{p}}$, we modify the sieve array as follows:

$$D[\nu(r + ka_\mathfrak{p})] \leftarrow D[\nu(r + ka_\mathfrak{p})] + \deg a_\mathfrak{p}$$

for all $k \in K[x]$ satisfying $0 \leq \deg(r + ka_\mathfrak{p}) \leq M$. We use the algorithm of Flassenberg and Paulus for efficiently jumping through the sieve array [FP99, §4.3.3]. Once this process has been completed for all $\mathfrak{p} \in \mathcal{F}_B$, each entry $D[\nu(j)] > T$ for some chosen **sieve tolerance value** T, corresponds to a candidate ideal $\mathfrak{b}_j$ with $\mathrm{N}(\mathfrak{b}_j) = g_0(j)$ that should be checked for smoothness.

Now we must discuss how to find $b_{\mathfrak{b}_j}$ for a candidate primitive ideal $\mathfrak{b}_j = (a_{\mathfrak{b}_j}, b_{\mathfrak{b}_j})$ with $\mathrm{N}(\mathfrak{b}_j) = a_{\mathfrak{b}_j} = g_0(j)$. Consider the **binary quadratic form** $F(u,v) = Au^2 + Buv + Cv^2$ for some $A, B, C \in K[x]$. These forms are useful because there is an map from binary quadratic forms to a basis of a primitive $\mathcal{O}$-ideal given by

$$\phi : Au^2 + Buv + Cv^2 \mapsto \{A, B + y\} .$$

One can obtain an **equivalent** form to F under the change of variables

$$\begin{pmatrix} u \\ v \end{pmatrix} = X \begin{pmatrix} U \\ V \end{pmatrix}, \qquad \text{where} \quad X = \begin{pmatrix} r & t \\ s & w \end{pmatrix} \in \mathrm{GL}_2(K[x]) .$$

Then $u = rU + tV$ and $v = sU + wV$ and

$$\begin{aligned} F(U,V) &= A(rU + tV)^2 + B(rU + tV)(sU + wV) + C(sU + wV)^2 \\ &= A(r^2U^2 + 2rtUV + t^2V^2) + B(rsU^2 + rwUV + stUV + twV^2) \\ &\quad + C(s^2U^2 + 2swUV + w^2V^2) \\ &= (Ar^2 + Brs + Cs^2)U^2 + \big(2(Art + Csw) + B(rw + st)\big)UV \\ &\quad + (At^2 + Btw + Cw^2)V^2 \\ &= F(r,s)U^2 + \big(2(Art + Csw) + B(rw + st)\big)UV + F(t,w)V^2 . \end{aligned}$$

Therefore, under the map ϕ there is an ideal with norm $F(r,s)$ [Jac99, pp. 25–26]. Given that the sieve polynomial $g_0(u) = au^2 + (2b - h)u + c$ is a binary quadratic form with the second indeterminant 1, we must have $r = j$, $s = 1$, and t, w satisfying $jw - t = \pm 1$. We choose $w = 0$ and $t = -1$. Then we have an ideal $\mathfrak{b}_j = (a_{\mathfrak{b}_j}, b_{\mathfrak{b}_j})$

under the map ϕ with norm $a_{\mathfrak{b}_j} = g_0(j)$ with $b_{\mathfrak{b}_j} = -2aj - (2b - h) = -2(aj + b) + h$.

For a candidate $\mathfrak{b}_j = (a_{\mathfrak{b}_j}, b_{\mathfrak{b}_j})$ with $a_{\mathfrak{b}_j} = g_0(j)$, suppose the smoothness test (Algorithm 4.10) returns a non-zero vector $\mathbf{w} = (w_1, w_2, \ldots, w_{n_B})$ satisfying $g_0(j) = \prod_{i=1}^{n_B} a_{\mathfrak{p}_i}^{w_i}$ for $\mathfrak{p}_i = (a_{\mathfrak{p}_i}, b_{\mathfrak{p}_i}) \in \mathcal{F}_B$. Similar to the baby walk method, we have a B-smooth ideal $(s_{\mathfrak{b}_j}, a_{\mathfrak{b}_j}, b_{\mathfrak{b}_j})$, where the coefficient $s_{\mathfrak{b}_j} \in \mathrm{Quot}(K[x])$ is computed from the inverses of the prime ideal factors as

$$s_{\mathfrak{b}_j} = \sum_{i:\substack{w_i \in \mathbf{w} \\ w_i < 0}} \left(\frac{1}{\mathrm{N}(\mathfrak{p}_i)}\right)^{|w_i|} = \sum_{i:\substack{w_i \in \mathbf{w} \\ w_i < 0}} \frac{1}{a_{\mathfrak{p}_i}^{|w_i|}} .$$

In Section 5.1.2 we describe precisely how we choose the ideal $\mathfrak{a}$ for the sieve polynomial. For now, suppose $\mathfrak{a} = (a_\mathfrak{a}, b_\mathfrak{a})$ is B-smooth, constructed such that $\mathrm{N}(\mathfrak{a}) = a_\mathfrak{a} = \prod_{i=1}^{n_B} \mathfrak{p}_i^{|v_i|}$ for a vector $\mathbf{v} = (v_1, \ldots, v_{n_B})$. Then we have a similar calculation for the coefficient $s_\mathfrak{a}$ based on the negative entries of $\mathbf{v}$:

$$s_\mathfrak{a} = \sum_{i:\substack{v_i \in \mathbf{v} \\ v_i < 0}} \left(\frac{1}{\mathrm{N}(\mathfrak{p}_i)}\right)^{|v_i|} = \sum_{i:\substack{v_i \in \mathbf{v} \\ v_i < 0}} \frac{1}{a_{\mathfrak{p}_i}^{|v_i|}} .$$

Now we have a relation for the primitive B-smooth ideal $(s_\mathfrak{a} s_{\mathfrak{b}_j} \alpha)\mathcal{O}$ consisting of the vector $\mathbf{v} + \mathbf{w}$ and the distance

$$\delta = \deg \alpha + \deg s_\mathfrak{a} + \deg s_\mathfrak{b} = \deg a_\mathfrak{a} + \deg j + \deg s_\mathfrak{a} + \deg s_\mathfrak{b} .$$

Note that both $\deg s_\mathfrak{a} \leq 0$ and $\deg s_\mathfrak{b} \leq 0$. We give the sieving method for generating relations in Algorithm 5.2.

Algorithm 5.2 (Sieving). *Generates relations for index calculus using sieving.*

Input: The factor base $\mathcal{F}_B = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_{n_B}\}$, a sieving polynomial $g_0(u)$, the ideal $\mathfrak{a} = (a_\mathfrak{a}, b_\mathfrak{a})$ used to generate $g_0(u)$, the vector $\mathbf{v} = (v_1, \ldots, v_{n_B})$ such that $\mathfrak{a} = \prod_{i=1}^{n_B} \mathfrak{p}_i^{v_i}$, and a set of roots $\mathcal{S}_{g_0,\mathfrak{p}}$ for each $\mathfrak{p} \in \mathcal{F}_B$. The sieving interval M, a sieve array D, and the tolerance value T.

Output: A set of relations $\mathcal{R}_B$.

1: $D = \{0, \ldots, 0\}$ ▷ Initialize the sieve array
2: **for each** $\mathfrak{p} = (a_\mathfrak{p}, b_\mathfrak{p}) \in \mathcal{F}_B$ **do**
3: **for each** $r \in \mathcal{S}_{g_0,\mathfrak{p}}$ **do** ▷ There are at most two roots in $\mathcal{S}_{g,\mathfrak{p}}$
4: $j \leftarrow r$, $k \leftarrow 0$
5: **while** $\deg(j) \leq M$ **do** ▷ Jump through indicies $\nu(r + ka_\mathfrak{p})$ for $k \in K[x]$
6: $D[\nu(j)] \leftarrow D[\nu(j)] + \deg a_\mathfrak{p}$
7: $k' \leftarrow \lambda(k)$ ▷ Compute the next multiplier (Alg. 5.1)
8: $e \leftarrow \deg(k' - k)$
9: $j \leftarrow j + a_\mathfrak{p} \cdot x^e$ ▷ Compute $r + ka_\mathfrak{p}$
▷ $a_\mathfrak{p} \cdot x^e$ implemented as shifting the coefficients of $a_\mathfrak{p}$ by e places
10: $k \leftarrow k'$
11: $k \leftarrow 0$, $d_\mathfrak{a} \leftarrow 0$, $d_\mathfrak{b} \leftarrow 0$, $\mathcal{R}_B \leftarrow \{\emptyset\}$
12: **while** $\deg(k) \leq M$ **do**
13: **if** $D[\nu(k)] > T$ **then** ▷ Determine candidates from the sieve array
14: $\mathfrak{b}_k \leftarrow \big(g_0(k), -2(ka_\mathfrak{a} + b_\mathfrak{a}) + h\big)$ ▷ Standard repr. of the candidate ideal
15: $\mathbf{w} \leftarrow \sigma(\mathcal{F}_B, \mathfrak{b}_k)$ ▷ Test the ideal $\mathfrak{b}_k$ for smoothness (Alg. 4.10)
16: **if** $\mathbf{w} \neq \mathbf{0}$ **then**
17: **for each** non-zero $w_i \in \mathbf{w}$ **do**

18: **if** $v_i < 0$ **then**

19: $d_{\mathfrak{a}} \leftarrow d_{\mathfrak{a}} - |v_i| \cdot \deg a_{\mathfrak{p}_i}$ ▷ Compute the degree of $s_{\mathfrak{a}}$

20: **if** $w_i < 0$ **then**

21: $d_{\mathfrak{b}} \leftarrow d_{\mathfrak{b}} - |w_i| \cdot \deg a_{\mathfrak{p}_i}$ ▷ Compute the degree of $s_{\mathfrak{b}}$

22: $\delta \leftarrow \deg a_{\mathfrak{a}} + \deg k + d_{\mathfrak{a}} + d_{\mathfrak{b}}$ ▷ The distance for the relation

23: $\mathcal{R}_B \leftarrow \mathcal{R}_B \cup \{(\mathbf{v} + \mathbf{w},\ \delta)\}$ ▷ Store the relation

24: $k \leftarrow \lambda(k)$ ▷ Compute the next index (Alg. 5.1)

Recall that in order to perform the linear algebra step in index calculus we require $m \geq n_B + 1$ relations. One method to obtain enough relations is to choose a large sieving inteval M; however, increasing M dramatically increases the running time of algorithm and the space it requires. We discuss a more practical alternative next.

5.1.2 Multiple Polynomial Sieving and Self-Initialization

It was first suggested by Montgomery in the context of the quadratic sieve factoring algorithm (*cf.* [Pom85, pp. 176–178]) that one could keep the sieve inteval M small by using **multiple sieving polynomials**. Therefore, if we do not acquire enough relations from the initial sieving polynomial, we choose a new sieving polynomial and repeat the process. To efficiently construct sieving polynomials of the form $g_\ell(u) = au^2 + (2b - h)u + c$, we choose a primitive $\mathcal{O}$-ideal $\mathfrak{a}_\ell = (a, b)$ according to a **self-initialization** technique. Self-initalization was introduced by Alford and Pomerance for factoring [AP95, §5]. Adapted from Jacobson in the number field case [Jac99, pp. 52–57], Velichka implemented self-initialization in the ideal class group of imaginary quadratic function fields [Vel08, pp. 80–84]. This applies directly in real

quadratic function fields. We generalize our discussion below from Velichka's to apply to both even and odd characteristic fields.

First we choose a subset of prime ideals $\mathcal{Q} = \{\mathfrak{q}_1, \mathfrak{q}_2, \ldots, \mathfrak{q}_{n_Q}\} \subset \mathcal{F}_B$ and let $\mathbf{v} = (v_1, v_2, \ldots, v_{n_Q})$. Note that by varying the $v_i \in \{1, -1\}$ we can obtain 2^{n_Q} possible sieve ideals that are of the form

$$(s_{a_\ell})\mathfrak{a}_\ell = \prod_{i=1}^{n_Q} \mathfrak{q}_i^{v_i}, \tag{5.1}$$

where we drop the coefficient s_{a_ℓ} (the coefficient s_{a_ℓ} is taken care of for each relation as described in the last section). In fact, to have useful relations we do not use a vector $\mathbf{v}$ if $-\mathbf{v}$ has already been chosen. This can be easily accomplished by forcing $v_{n_Q} = 1$, thus giving 2^{n_Q-1} possible ideals. There are two advantages to using such ideals. One is that we can easily obtain a new sieving polynomial without computing Equation (5.1) each time. The second advantage is that there is an efficient way to find the roots $\mathcal{S}_{g_\ell,\mathfrak{p}}$ for the new sieving polynomial g_ℓ. This can be seen in Algorithm 5.4 after having performed self-initialization as in Algorithm 5.3. These algorithms were adapted from Velichka [Vel08, pp. 82–84] to support both even and odd field characteristics.

Algorithm 5.3 (Setup for Self-initialization). *Chooses an ideal $\mathfrak{a}_0$ and computes an initial sieving polynomial g_0 along with a set of roots $\mathcal{S}_{g_0,\mathfrak{p}}$ for each $\mathfrak{p} \in \mathcal{F}_B$ according to the self-initialization technique.*

Input: The real hyperelliptic curve equation $C : y^2 + h(x)y = f(x)$, the factor base $\mathcal{F}_B$ and a size parameter $n_Q < n_B$

Output: An initial sieving polynomial g_0, the ideal $\mathfrak{a}_0 = (a, b_0)$ used to generate g_0,

and $\mathcal{S}_{g_0,\mathfrak{p}}$ for each $\mathfrak{p} \in \mathcal{F}_B$.

1: $\mathcal{Q} = \{\mathfrak{q}_1, \ldots, \mathfrak{q}_{n_Q}\} \subset \mathcal{F}_B$ ▷ Select a subset $\mathcal{Q} \subset \mathcal{F}_B$ of size n_Q

2: $\mathbf{v} \leftarrow (1, \ldots, 1)$ ▷ Select the initial exponent vector

3: $\mathfrak{a}_0 = (a, b_0) \leftarrow \prod_{i=1}^{n_Q} \mathfrak{q}_i$ ▷ Compute b_0

4: $c_0 \leftarrow (b_0^2 - hb_0 - f)/a$

5: $g_0 \leftarrow au^2 + (2b_0 - h)u + c_0$ ▷ Compute the initial sieving polynomial

6: **for each** $\mathfrak{q}_i = (a_{\mathfrak{q}_i}, b_{\mathfrak{q}_i}) \in \mathcal{Q}$ **do**

7: $\quad B_i \leftarrow (a/a_{\mathfrak{q}_i})\big((a/a_{\mathfrak{q}_i})^{-1} b_{\mathfrak{q}_i} \mod a_{\mathfrak{q}_i}\big) \mod a$

8: $\quad \overline{B}_i \leftarrow (a/a_{\mathfrak{q}_i})\big((a/a_{\mathfrak{q}_i})^{-1}(-b_{\mathfrak{q}_i} - h) \mod a_{\mathfrak{q}_i}\big) \mod a$

9: $\quad \widehat{B}_i \leftarrow (B_i + \overline{B}_i)/a$

10: **for each** $\mathfrak{p} = (a_\mathfrak{p}, b_\mathfrak{p}) \in \mathcal{F}_B$ **do** ▷ Find the roots of $g_0 \bmod a_\mathfrak{p}$

11: $\quad$ **if** $a_\mathfrak{p} | a \ \wedge \ a_\mathfrak{p} | (2b_0 - h)$ **then** ▷ $g_0(u) \equiv c_0 \pmod{a_\mathfrak{p}}$

12: $\quad\quad \mathcal{S}_{g_0,\mathfrak{p}} \leftarrow \{\emptyset\}$ ▷ No roots

13: $\quad$ **else if** $a_\mathfrak{p} | a \ \wedge \ a_\mathfrak{p} \nmid (2b_0 - h)$ **then** ▷ $g_0(u) \equiv (2b_0 - h)u + c_0 \pmod{a_\mathfrak{p}}$

14: $\quad\quad \mathcal{S}_{g_0,\mathfrak{p}} \leftarrow \{-c_0(2b_0 - h)^{-1} \bmod a_\mathfrak{p}\}$ ▷ One root

15: $\quad$ **else if** $a_\mathfrak{p} | f$ **then** ▷ $g_0(u) \equiv au^2 + (2b_0 - h)u + (b_0^2 - hb_0)a^{-1} \pmod{a_\mathfrak{p}}$

16: $\quad\quad \mathcal{S}_{g_0,\mathfrak{p}} \leftarrow \{-b_0 a^{-1} \bmod a_\mathfrak{p},\ (h - b_0)a^{-1} \bmod a_\mathfrak{p}\}$

17: $\quad$ **else**

18: $\quad\quad$ Find the two roots r_1, r_2 of $g_0(u) \bmod a_\mathfrak{p}$ ▷ See remarks below

19: $\quad\quad \mathcal{S}_{g_0,\mathfrak{p}} \leftarrow \{r_1,\ r_2\}$

In Step 18 of Algorithm 5.3 we are required to find the roots of a quadratic polynomial of the form $y^2 + by + c$, with $b, c \in K[x]$, modulo an irreducible polynomial $p \in K[x]$. Algorithms for performing this are discussed in Section 4.3.2.

Once one has performed the setup for self-initialization using Algorithm 5.3, new

sieving polynomials can be found using Algorithm 5.4.

Algorithm 5.4 (New polynomial from self-initialization). *Computes a sieving polynomial g_ℓ from a new ideal $\mathfrak{a}_\ell$ along with a set of roots $\mathcal{S}_{g_\ell,\mathfrak{p}}$ for each $\mathfrak{p} \in \mathcal{F}_B$ after Algorithm 5.3 has performed self-initialization.*

Input: The real hyperelliptic curve equation $C : y^2+h(x)y = f(x)$, the factor base $\mathcal{F}_B$ and subset $\mathcal{Q} \subset \mathcal{F}_B$ of size n_Q. The number ℓ of polynomials previously obtained, the polynomial $b_{\ell-1}$ from the previous ideal $\mathfrak{a}_{\ell-1} = (a, b_{\ell-1})$, roots $\mathcal{S}_{g_{\ell-1},\mathfrak{p}}$ for each $\mathfrak{p} \in \mathcal{F}_B$, the previously used vector $\mathbf{v}$, and the three sets of values $\{B_1, \ldots, B_{n_Q}\}$, $\{\overline{B}_1, \ldots, \overline{B}_{n_Q}\}$ and $\{\widehat{B}_1, \ldots, \widehat{B}_{n_Q}\}$

Output: A new sieving polynomial g_ℓ, the polynomial b_ℓ from the new ideal $\mathfrak{a}_\ell = (a, b_\ell)$ used to generate g_ℓ, and $\mathcal{S}_{g_\ell,\mathfrak{p}}$ for each $\mathfrak{p} \in \mathcal{F}_B$.

1: $\ell \leftarrow \ell \pmod{2^{n_Q-1}}$
2: $k \leftarrow \lfloor \log_2(\ell - 1) \rfloor + 1$ ▷ Choose which exponent to flip
3: $\mathbf{v} \leftarrow (v_1, \ldots, v_{k-1}, -v_k, v_{k+1}, \ldots, v_{n_Q})$ ▷ Update the exponent vector
4: $b_\ell \leftarrow b_{\ell-1} + B_k + \overline{B}_k$ ▷ The new ideal is $\mathfrak{a}_\ell = (a, b_\ell)$
5: $c_\ell \leftarrow (b_\ell^2 - b_\ell h - f)/a$
6: $g_\ell \leftarrow au^2 + (2b_\ell - h)u + c_\ell$ ▷ Compute the new sieving polynomial
7: **for each** $\mathfrak{p} = (a_\mathfrak{p}, b_\mathfrak{p}) \in \mathcal{F}_B$ **do** ▷ Find the roots of $g_\ell \bmod a_\mathfrak{p}$
8: **if** $a_\mathfrak{p} | a \ \wedge \ a_\mathfrak{p} | (2b_\ell - h)$ **then** ▷ $g_\ell(u) \equiv c_\ell \pmod{a_\mathfrak{p}}$
9: $\mathcal{S}_{g_\ell,\mathfrak{p}} \leftarrow \{\emptyset\}$ ▷ No roots
10: **else if** $a_\mathfrak{p} | a \ \wedge \ a_\mathfrak{p} \nmid (2b_\ell - h)$ **then** ▷ $g_\ell(u) \equiv (2b_\ell - h)u + c_\ell \pmod{a_\mathfrak{p}}$
11: $\mathcal{S}_{g_\ell,\mathfrak{p}} \leftarrow \{-c_\ell(2b_\ell - h)^{-1} \bmod a_\mathfrak{p}\}$ ▷ One root
12: **else**
13: $\mathcal{S}_{g_\ell,\mathfrak{p}} \leftarrow \left\{ r - \widehat{B}_k \bmod a_\mathfrak{p} \;\middle|\; r \in \mathcal{S}_{g_{\ell-1},\mathfrak{p}} \right\}$ ▷ Two roots

Step 13 of Algorithm 5.4 shows how self-initialization allows one to compute the roots of the new sieve polynomial extremely quickly. For each $\mathfrak{p} \in \mathcal{F}_B$, the test whether $a_\mathfrak{p} | a$ remains constant for every sieve polynomial. Therefore, those $\mathfrak{p}$ for which $g_{\ell-1}$ has two roots modulo $a_\mathfrak{p}$ remains the same for g_ℓ. However, those $\mathfrak{p}$ for which $g_{\ell-1}$ has one root or no roots modulo $a_\mathfrak{p}$ may swap for g_ℓ if the characteristic is odd. Thus in Step 13 where we had two roots $r_1, r_2 \in \mathcal{S}_{g_{\ell-1}}$ for $g_{\ell-1} \bmod a_\mathfrak{p}$, we compute the roots for the new polynomial $g_\ell \bmod a_\mathfrak{p}$ by subtracting $\widehat{B}_k = (B_k + \overline{B}_k)/a$ from each of r_1 and r_2. Since $b_\ell - b_{\ell-1} = B_k + \overline{B}_k$ from Step 4, we can show that if r is a root of $g_{\ell-1} \bmod a_\mathfrak{p}$, then $r - (B_k + \overline{B}_k)a^{-1}$ is indeed a root of $g_\ell \bmod a_\mathfrak{p}$:

$$
\begin{aligned}
& g_\ell\big(r - (B_k + \overline{B}_k)a^{-1}\big) \\
&= a\big(r - (B_k + \overline{B}_k)a^{-1}\big)^2 + (2b_\ell - h)\big(r - (B_k + \overline{B}_k)a^{-1}\big) + (b_\ell^2 - b_\ell h - f)a^{-1} \\
&= ar^2 - 2(B_k + \overline{B}_k)r + (B_k + \overline{B}_k)^2 a^{-1} \\
&\qquad + \big(2(B_k + \overline{B}_k + b_{\ell-1}) - h\big)\big(r - (B_k + \overline{B}_k)a^{-1}\big) \\
&\qquad + \big((B_k + \overline{B}_k + b_{\ell-1})^2 - (B_k + \overline{B}_k + b_{\ell-1})h - f\big)a^{-1} \\
&= ar^2 - 2(B_k + \overline{B}_k)r + (B_k + \overline{B}_k)^2 a^{-1} \\
&\qquad + (2b_{\ell-1} - h)r - 2(B_k + \overline{B}_k)b_{\ell-1}a^{-1} + (B_k + \overline{B}_k)ha^{-1} + 2(B_k + \overline{B}_k)r - 2(B_k + \overline{B}_k)^2 a^{-1} \\
&\qquad + (b_{\ell-1}^2 - b_{\ell-1}h - f)a^{-1} + 2(B_k + \overline{B}_k)b_{\ell-1}a^{-1} + (B_k + \overline{B}_k)^2 a^{-1} - (B_k + \overline{B}_k)ha^{-1} \\
&= ar^2 + (2b_{\ell-1} - h)r + (b_{\ell-1}^2 - b_{\ell-1}h - f)a^{-1} \\
&= g_{\ell-1}(r) \equiv 0 \pmod{a_\mathfrak{p}} .
\end{aligned}
$$

Although it is difficult to rigourously analyze, sieving has been shown to be more efficient than random walks in practice (*cf.* Velichka [Vel08, pp. 111–120]). For our implementation results we compared sieving with our baby walk strategy.

5.1.3 Low-Degree Sieving

In practice we consider another parameter for sieving by choosing a **sieve bound** $S \leq B$. Since an increase in the smoothness bound B typically has a large effect on the size of the factor base $\mathcal{F}_B$, the idea is that we sieve with a subset $\mathcal{F}_S \subseteq \mathcal{F}_B$, where

$$\mathcal{F}_S = \left\{ \mathfrak{p} \in \mathcal{F}_B \;\middle|\; \deg \mathfrak{p} \leq S \right\}.$$

That is, the following lines of the sieve algorithms would have $\mathcal{F}_B$ replaced by $\mathcal{F}_S$: Algorithm 5.2, Step 2; Algorithm 5.3, Step 10; and Algorithm 5.4, Step 7. We still perform the smoothness test on the candidates with the full factor base $\mathcal{F}_B$. Therefore, we must reduce the tolerance value T appropriately to still obtain candidates that factor over the primes of degree B.

This technique was previously discussed by Velichka for sieving in imaginary quadratic function fields [Vel08, pp. 79–80].

5.1.4 Large Primes and Partial Relations

Another technique for improving the relation generation step of index calculus is to allow the use of large primes. This has been used effectively in integer factorization by Morrison and Brillhart [MB75, §4] and described in the context of discrete logarithms in $(\mathbb{Z}/q\mathbb{Z})^*$ by Odlyzko [Odl85, §5.4]. Jacobson extended large primes to quadratic number fields [Jac99, pp. 58–61], and it was implemented in characteristic-2 imaginary quadratic function fields by Velichka [Vel08, pp. 62–64]. We describe the technique as applied to the baby walk method in the infrastructure, but it can be easily adapted to sieving in our setting as well.

The idea of using large primes is that if an ideal tested for smoothness has one prime factor that is larger than those in the factor base, we store the factorization along with the large prime separately as a "partial relation." If we find two partial relations corresponding to the same prime ideal, we combine the two relations to remove the large prime, thus obtaining a regular "full relation" that we can add to $\mathcal{R}_B$.

Let $\mathcal{F}_B = \{\mathfrak{p}_1, \mathfrak{p}_2, \ldots, \mathfrak{p}_{n_B}\}$ be a factor base of prime ideals. Suppose in addition to the set of relations $\mathcal{R}_B$, we keep another set $\mathcal{P}_{B_L}$ consisting of **partial relations** of the form $\left(\mathfrak{q}_j, s_j, \mathbf{e}_j = (e_1, \ldots, e_{n_B}), \delta_j\right)$ satisfying

$$\mathfrak{q}_j^{s_j} \cdot \prod_{k=1}^{n_B} \mathfrak{p}_k^{e_k} = \alpha\mathcal{O} \qquad \text{for some } \alpha \in F \text{ such that} \qquad \delta_j = \deg \alpha\,, \tag{5.2}$$

where $\mathfrak{q}_j$ is some $\mathcal{O}$-ideal with irreducible norm that is not in the factor base $\mathcal{F}_B$ and $s_j = \pm 1$. We call $\mathfrak{q}_j$ a **large prime** since its norm is irreducible over $K[x]$ and it must satisfy $\deg \mathrm{N}(\mathfrak{q}_j) > B$ if it was not included in $\mathcal{F}_B$. A large prime $\mathfrak{q}_j$ included in the set of partial relations $\mathcal{P}_{B_L}$ will satisfy $B < \deg \mathrm{N}(\mathfrak{q}_j) \leq B_L$ for some bound $B_L > B$. We call the ideal $\alpha\mathcal{O}$ in Equation (5.2) for which we found the relation **almost B-smooth**. To avoid confusion, we call relations in $\mathcal{R}_B$ **full relations** [AT06, pp. 507–508].

One can obtain a full relation from partial relations as follows. Suppose we have two partial relations in $\mathcal{P}_{B_L}$ with large primes $\mathfrak{q}_i = \mathfrak{q}_j$. Then if $s_i = s_j$ we have a full relation $(\mathbf{e}, \delta)$ given by $\mathbf{e} = \mathbf{e}_i - \mathbf{e}_j$ and $\delta = \delta_i - \delta_j$. Otherwise, if $s_i = -s_j$, then the full relation is computed as $\mathbf{e} = \mathbf{e}_i + \mathbf{e}_j$ and $\delta = \delta_i + \delta_j$. Note that in both cases the large prime in Equation (5.2) cancels.

The remaining question is how to find these partial relations. Essentially, we

perform the smoothness test as in Algorithm 4.10, but we test up to the bound B_L. If the polynomial factorization results in roots that are all contained in $\mathcal{F}_B$, then the ideal is B-smooth. If the polynomial roots are all in $\mathcal{F}_B$ except for one with degree d such that $B < d \leq B_L$, then the ideal is almost B-smooth. Otherwise, we consider the tested ideal to be neither smooth nor almost smooth. This change is reflected in Algorithm 5.5.

Algorithm 5.5 (Smoothness test σ). *Test whether a reduced ideal is smooth or almost smooth with a possible coefficient and returns the relation vector if it is.*

Input: The order q of the field K, a factor base $\mathcal{F}_B = \{\mathfrak{p}_1 = (a_{\mathfrak{p}_1}, b_{\mathfrak{p}_1}), \ldots, \mathfrak{p}_{n_B} = (a_{\mathfrak{p}_{n_B}}, b_{\mathfrak{p}_{n_B}})\}$, a large prime bound B_L, and a primitive $\mathcal{O}$-ideal $\mathfrak{a} = (a_\mathfrak{a}, b_\mathfrak{a})$.

Output: A vector $\mathbf{e} = (e_1, \ldots, e_{n_B})$ and a prime ideal $\mathfrak{q} \notin \mathcal{F}_B$ with an exponent $s \in \{0, \pm 1\}$ such that $\mathfrak{q}^s \prod_{i=1}^{n_B} \mathfrak{p}_i^{e_i} = \mathfrak{a}$ if $\mathfrak{a}$ is B-smooth or almost B-smooth; or $\mathbf{e} = \mathbf{0}$ with $\mathfrak{q} = \mathcal{O}$ and $s = 0$ otherwise.

1: $a \leftarrow \mathsf{Squarefree}(a_\mathfrak{a})$ ▷ Compute the squarefree part of $a_\mathfrak{a}$
2: $\xi \leftarrow \Phi^{\lfloor \frac{t}{2} \rfloor}(x) \pmod a$ ▷ Iterative Frobenius map (Alg. 4.9)
3: **for** i **from** $\lfloor \frac{t}{2} \rfloor + 1$ **to** B **do** ▷ Perform distinct degree factorization
4: $\quad \xi \leftarrow \Phi(\xi) \pmod a$ ▷ Frobenius map
5: $\quad d \leftarrow \gcd(\xi - x, a)$
6: $\quad a \leftarrow a/d$
7: **if** a is not irreducible $\wedge$ $\deg a > B_L$ **then** ▷ Not B-smooth/almost B-smooth
8: $\quad \mathbf{e} \leftarrow (0, \ldots, 0)$, $\mathfrak{q} = \mathcal{O}$, $s = 0$
9: **else**
10: $\quad p_1^{n_1} p_2^{n_2} \cdots p_\ell^{n_\ell} \leftarrow \mathsf{PolyFactor}(a_\mathfrak{a})$ ▷ Compute the factors of $a_\mathfrak{a}$
11: $\quad e_1, e_2, \ldots, e_{n_B} \leftarrow 0$, $\mathcal{S} \leftarrow \emptyset$ ▷ Initialize the e_i to 0

12: **for** i **from** 1 **to** ℓ **do**
13: Find k such that $p_i = a_{\mathfrak{a}_k}$ for $\mathfrak{p}_k \in \mathcal{F}_B$
14: **if** no k found **then**
15: $\mathcal{S} \leftarrow \mathcal{S} \cup \{(p_i, n_i)\}$ ▷ Temporarily store the large prime
16: **else**
17: $e_k \leftarrow n_i$
18: **if** $b_{\mathfrak{a}} \not\equiv b_{\mathfrak{p}_k} \pmod{p_i}$ **then**
19: $e_k \leftarrow -e_k$ ▷ Correct the sign of the exponent
20: **if** $\mathcal{S} = \emptyset$ **then** ▷ $\mathfrak{a}$ is B-smooth
21: $\mathbf{e} \leftarrow (e_1, e_2, \ldots, e_{n_B})$, $\mathfrak{q} = \mathcal{O}$, $s = 1$
22: **else if** $\mathcal{S} = \{(p, n)\}$ for some p **then** ▷ $\mathfrak{a}$ is almost B-smooth
23: Find a root b of $y^2 + hy - f \pmod{p}$
24: **if** $b_{\mathfrak{a}} \not\equiv b \pmod{p}$ **then**
25: $s \leftarrow -n$ ▷ Correct the sign of the exponent
26: **else**
27: $s \leftarrow n$
28: $\mathbf{e} \leftarrow (e_1, e_2, \ldots, e_{n_B})$, $\mathfrak{q} \leftarrow (p, b)$
29: **else** ▷ $\mathfrak{a}$ is not B-smooth or almost B-smooth
30: $\mathbf{e} \leftarrow (0, \ldots, 0)$, $\mathfrak{q} = \mathcal{O}$, $s = 0$

For computing a root of a quadratic polynomial modulo p in Step 23 of Algorithm 5.5, we refer to the discussion in Section 4.3.2. The asymptotic complexity is unchanged from Algorithm 4.10, requiring $O(g^3 \log^2 q + g^2 \log^3 q)$ bit operations.

For a non-empty set of partial relations $\mathcal{P}_{B_L}$, we find matches and combine partial relations into full relations using Algorithm 5.6. The set $\mathcal{P}_{B_L}$ is often quite large,

thus requiring one to write $\mathcal{P}_{B_L}$ to a file and sort before searching for matches. By estimating the number of partial relations that will become full relations, we can perform Algorithm 5.6 just once at the end to reduce the number of times the file is read in and sorted [Vel08, p. 64]. We calculate such an estimation in Section 5.3.1. Let n_{B_L} be an estimate of the number of large primes. We will see in Theorem 5.9 that we expect to get $\lfloor \#\mathcal{P}_{B_L}/n_{\mathcal{P}} \rfloor$ full relations from combining the set $\mathcal{P}_{B_L}$ of partial relations, where $n_{\mathcal{P}} = \sqrt{\frac{3}{2}n_{B_L}}$. Therefore, we call Algorithm 5.6 to combine the partial relations once the number of full relations obtained is $\#\mathcal{R}_B \geq m - \lfloor \#\mathcal{P}_{B_L}/n_{\mathcal{P}} \rfloor$, where m is the total number of relations desired.

Algorithm 5.6 (Partial relation combining). *Combine matching partial relations into full relations.*

Input: A set $\mathcal{R}_B$ of previously generated full relations, and a set $\mathcal{P}_{B_L}$ of previously generated partial relations.

Output: A set $\mathcal{R}_B$ of full relations and a set $\mathcal{P}_{B_L}$ of partial relations.

1: Sort $\mathcal{P}_{B_L}$ by the large primes
2: **for each** pair $(\mathfrak{q}, s_1, \mathbf{e}_1, \delta_1), (\mathfrak{q}, s_2, \mathbf{e}_2, \delta_2) \in \mathcal{P}_{B_L}$ **do**
3: **if** $s_1 = s_2$ **then** ▷ Combine the partials into a full relation
4: $\mathbf{e} = (e_1, \ldots, e_{n_B}) \leftarrow \mathbf{e}_1 - \mathbf{e}_2$
5: $\delta \leftarrow \delta_1 - \delta_2$
6: **else**
7: $\mathbf{e} = (e_1, \ldots, e_{n_B}) \leftarrow \mathbf{e}_1 + \mathbf{e}_2$
8: $\delta \leftarrow \delta_1 + \delta_2$
9: **if** $s_1 < 0 \ \wedge s_2 < 0$ **then** ▷ Adjust the distance for the large prime
10: $\delta \leftarrow \delta - 2 \deg \mathfrak{q}$

11: **else if** $s_1 < 0 \ \vee s_2 < 0$ **then**

12: $\quad \delta \leftarrow \delta - \deg \mathfrak{q}$

13: $\mathcal{R}_B \leftarrow \mathcal{R}_B \cup \{(\mathfrak{e}, \delta)\}$ ▷ Store the full relation

14: $\mathcal{P}_{B_L} \leftarrow \mathcal{P}_{B_L} \setminus \{(\mathfrak{q}, s_2, \mathfrak{e}_2, \delta_2)\}$ ▷ Remove one of the matching partials

We also update the baby walk relation generation from Algorithm 4.12 to be able to handle almost-smooth ideals. So that this method is effective with large primes, we forego computing a strictly diagonally dominant matrix and just use the baby walk starting from $\mathcal{O}$ (*cf.* Section 5.2). This change is reflected in Algorithm 5.7. We omit an updated sieving method for relation generation since the modifications from Algorithm 5.2 are similar to those presented in Algorithm 5.7.

Algorithm 5.7 (Relation generation). *Generate a relation for index calculus in the infrastructure using a baby walk and large primes.*

Input: A factor base $\mathcal{F}_B = \{\mathfrak{p}_1 = (a_{\mathfrak{p}_1}, b_{\mathfrak{p}_1}), \ldots, \mathfrak{p}_{n_B} = (a_{\mathfrak{p}_{n_B}}, b_{\mathfrak{p}_{n_B}})\}$, a set $\mathcal{R}_B$ of previously generated full relations, and a set $\mathcal{P}_{B_L}$ of previously generated partial relations. If $\#\mathcal{R}_B \neq 0$, then we require the $\mathcal{O}$-ideal $\mathfrak{a}$ used to generate the last relation and its distance $\delta(\mathfrak{a})$.

Output: A set $\mathcal{R}_B$ of full relations and a set $\mathcal{P}_{B_L}$ of partial relations.

1: $k \leftarrow \#\mathcal{R}_B + 1$, $j \leftarrow 0$

2: **if** $k = 1$ **then**

3: $\quad \mathfrak{a}_0 \leftarrow \mathcal{O}$, $\delta_j \leftarrow 0$

4: **else**

5: $\quad \mathfrak{a}_0 \leftarrow \mathfrak{a}$, $\delta_j \leftarrow \delta(\mathfrak{a})$

6: $(\mathfrak{a}_{j+1}, \delta_{j+1}) \leftarrow \rho(\mathfrak{a}_j, \delta_j)$ ▷ Next in the baby walk (Alg. 3.10)

7: $j \leftarrow j + 1$

8: **repeat**

9: $(\mathbf{e}, \mathfrak{q}, s) \leftarrow \sigma(\mathcal{F}_B, \mathfrak{a}_j)$ ▷ Test if $\mathfrak{a}_j$ is smooth or almost-smooth (Alg. 5.5)

10: **if** $\mathbf{e} \neq \mathbf{0}$ **then**

11: $\delta \leftarrow \delta_j$

12: **for each** non-zero $e_i \in \mathbf{e}$ **do**

13: **if** $e_i < 0$ **then** ▷ Adjust the distance for inverses

14: $\delta \leftarrow \delta + e_i \cdot \deg a_{\mathfrak{p}_i}$

15: **if** $s = 0$ **then** ▷ $\mathfrak{a}_i$ is B-smooth

16: $\mathcal{R}_B \leftarrow \mathcal{R}_B \cup \{(\mathbf{e}, \delta)\}$ ▷ Store the full relation

17: **else** ▷ $\mathfrak{a}_j$ is almost B-smooth

18: $\mathcal{P}_{B_L} \leftarrow \mathcal{P}_{B_L} \cup \{(\mathfrak{q}, s, \mathbf{e}, \delta_j)\}$ ▷ Store the partial relation

19: **else** ▷ If $\mathbf{e} = \mathbf{0}$, *i.e.* $\mathfrak{a}_j$ is not smooth or almost smooth

20: $(\mathfrak{a}_{j+1}, \delta_{j+1}) \leftarrow \rho(\mathfrak{a}_j, \delta_j)$ ▷ Next in the baby walk (Alg. 3.10)

21: $j \leftarrow j + 1$

22: **until** $\mathbf{e} \neq \mathbf{0} \ \wedge \ \mathfrak{q} = \mathcal{O}$

Partial relations are obviously more plentiful than full relations. Consequently, it is expected that using large primes will reduce the number of walk steps required to find enough relations. We estimate this improvement in Section 5.3.1. However, a full relation resulting from combining partial relations is slightly less sparse, perhaps slowing down linear algebra. This is only significant if partial relations with multiple large primes are allowed (*cf.* Holt and Davenport [HD03]). Implementations of index calculus using multiple large primes were done by Holt and Davenport in $(\mathbb{Z}/q\mathbb{Z})^*$ and by Gaudry, Thomé, Thériault and Diem in low genus imaginary quadratic function

fields [GTTD07]. In the work at hand, we did not investigate multi-large prime variations.

5.1.5 Linear Algebra Improvements

The linear algebra techniques described in Chapter 4 are the best asymptotic results we are aware of. In practice, however, we can significantly improve on some of those methods.

To compute the regulator in Section 4.3.6, we suggested that one compute the Hermite normal form of the augmented relation matrix $A'_{\mathcal{R}}$, whose determinant will be the product $h_{\mathcal{O}}R_{\mathcal{O}}$. Firstly, the HNF is expensive to compute in practice. Secondly, the determinant is exactly equal to $h_{\mathcal{O}}R_{\mathcal{O}}$ only if the smoothness bound B is chosen to be large enough that the prime ideals in $\mathcal{F}_B$ generate the class group and the columns of $A'_{\mathcal{R}}$ generate the full lattice Λ'_B.

If we reduce the smoothness bound B, our relation matrix $A'_{\mathcal{R}}$ may only generate a sublattice of Λ'_B. Consequently, the determinant of the HNF of $A'_{\mathcal{R}}$ will be a multiple of $h_{\mathcal{O}}R_{\mathcal{O}}$. As long as this multiple is not too large, we can still use our factoring technique of Algorithm 4.16 to obtain the regulator $R_{\mathcal{O}}$. The disadvantage is that we are no longer able to find the class number from this computation.

For regulator computation we replaced the HNF computation with two different methods. The first technique we call the **determinant method**. Let A be an $(n_B + 1) \times (n_B + 1)$ submatrix of $A'_{\mathcal{R}}$ with the linearly dependent columns removed. Then the determinant of the matrix A will be some multiple of the regulator $R_{\mathcal{O}}$.

We can use a method based on Wiedemann's algorithm to remove the linearly dependent columns from $A'_{\mathcal{R}}$. Recall that Wiedemann's algorithm works over a field.

In practice, we suggest choosing a random word-sized prime p and use the field $L = \mathbb{Z}/p\mathbb{Z}$. Then to remove linearly dependent columns, we begin by choosing an $n_B \times n_B$ submatrix A consisting of the first n_B columns of $A'_{\mathcal{R}}$ along with another column in $A'_{\mathcal{R}}$ chosen for $\mathbf{b}$. Wiedemann's algorithm for solving $A\mathbf{x} = \mathbf{b} \pmod{p}$ will compute a solution $\mathbf{x}$ or determine a column dependency in A. If a solution $\mathbf{x}$ is found, then $\mathbf{b}$ is linearly dependent modulo p in $A'_{\mathcal{R}}$ since $\mathbf{x}$ gives a linear combination of the columns of A that result in $\mathbf{b}$. We can continue the process of replacing linearly dependent columns in $[A|\mathbf{b}]$ with the extra columns of $A'_{\mathcal{R}}$ until we find a square nonsingular matrix or an inconsistent system [Wie86, §III]. Since this method performs operations in the field L and takes advantage of the sparsity of the relation matrix, it is expected to be much faster than computing the HNF over the integers.

Wiedemann also gave a method for computing the determinant of a sparse $n_B \times n_B$ matrix in $O\big(n_B(\omega + n_B \log n_B)\big)$ field operations, where ω denotes the number of non-zero entries in the input matrix [Wie86, §V]. To compute the determinant over the integers we suggest performing Wiedemann's determinant algorithm over multiple fields of the form $L = \mathbb{Z}/p\mathbb{Z}$ for different randomly chosen word-sized primes. We combine the determinants using Chinese remainder theorem, repeating the process until the CRT result stabilizes.

The multiple of the regulator may be quite large in the determinant method, slowing down the factorization in Algorithm 4.16. One option to reduce this multiple is to repeat the determinant method after replacing a column of $A'_{\mathcal{R}}$ with a different linearly independent column and perform Algorithm 4.16 on the greatest common divisor of the two determinants. The result of the GCD is likely a small multiple of $R_{\mathcal{O}}$ since we expect the two determinants to be different multiples of $h_{\mathcal{O}} R_{\mathcal{O}}$. However,

we present another method which also results in a small multiple of the regulator, but requires fewer linear algebra computations.

We call our second technique for computing the regulator the **kernel method**. Cohen attributes the ideas of this method in the number field case to Buchmann [Coh93, p. 288]. Suppose we compute two kernel vectors $\mathbf{u}, \mathbf{v}$ of $A_{\mathcal{R}}$.. From $\mathbf{u} = (u_1, u_2, \ldots, u_{n_B})$ and $\mathbf{v} = (v_1, v_2, \ldots, v_{n_B})$ we compute the two distances $\delta_{\mathbf{u}}$ and $\delta_{\mathbf{v}}$ corresponding to

$$\delta_{\mathbf{u}} = \sum_{i=1}^{n_B} u_i \cdot \delta(\mathfrak{p}_i) \qquad \text{and} \qquad \delta_{\mathbf{v}} = \sum_{i=1}^{n_B} v_i \cdot \delta(\mathfrak{p}_i) \, .$$

Note that $\delta_{\mathbf{u}}$ and $\delta_{\mathbf{v}}$ must be both multiples of the regulator $R_{\mathcal{O}}$. Then we can get a small multiple of $R_{\mathcal{O}}$ (or often $R_{\mathcal{O}}$ itself) by computing $kR_{\mathcal{O}} = \gcd(\delta_{\mathbf{u}}, \delta_{\mathbf{v}})$. To find the actual regulator $R_{\mathcal{O}}$, we again use Algorithm 4.16.

To compute kernel vectors over $\mathbb{Z}$, we suggest using an algorithm by Eberly *et al.* [EGG+06, EGG+07].. While thoroughly analyzed asymptotically, there is also an implementation of the algorithm in the `LinBox` library [EGG+06, §4]. Specifically, this algorithm computes a rational solution $\mathbf{x}$ to the system $A_{\mathcal{R}}\mathbf{x} = \mathbf{b}$. To get a random kernel vector, we choose a vector $\mathbf{v}$ with random integer entries and compute $\mathbf{b} = A_{\mathcal{R}}\mathbf{v}$. Now, using the algorithm to solve $A_{\mathcal{R}}\mathbf{x} = \mathbf{b}$, we obtain a vector $\mathbf{x}$ which we correct to a kernel vector by subtracting $\mathbf{v}$. Note that if the entries of $\mathbf{x}$ are rational and not in $\mathbb{Z}$, then we must add new relations until we get an integer solution.

5.2 Implementation Details

To implement index calculus in the infrastructure of a real quadratic function field, we adapted an implementation of index calculus in imaginary quadratic function fields by Velichka (*cf.* [Vel08]). This was part of a larger C++ library for algebraic number theory called `ANTL` started by Jacobson (*cf.* [Jac99]). The library is based on Shoup's `NTL` library [Sho08]. We compiled `NTL` to use the `GMP` library [Gra07] for large integer arithmetic.

Degree Computations

One issue that arose in our implementation was the precision for computing the degree of a function field element $\alpha = (a + by)/d$, where $a, b, d \in K[x]$. The difficulty lies in the fact that the root y of the function field equation $\Psi(T) = T^2 + h(x)T - f(x) \in K[x][T]$ is an element of the Puiseux series $K((1/x))$ (*cf.* Section 3.2.1). That is, $y = \sum_{i=-\infty}^{m} c_i x^i \in K((1/x))$ with $c_m \neq 0$ for some $m \in \mathbb{N}_0$. Initially, we used m as the degree of y, *i.e.* $y \approx \lfloor y \rfloor = \sum_{i=0}^{m} c_i x_i$, and computed

$$\deg \alpha = \deg(a + b \cdot \lfloor y \rfloor) - \deg d \,.$$

However, our testing indicated a problem. We discovered that it was necessary to consider a sufficient number of negative-degree terms of y. Therefore, we represented the root as $y \approx x^k \lfloor y \rfloor / t \in \mathrm{Quot}(K[x])$ with $t = \sum_{i=1}^{k} c_{-i} x^i$ for some bound $k \in \mathbb{N}$. We chose $k = 2 \cdot \max\{\deg a,\ \deg b\}$ and computed the degree as

$$\deg \alpha = \deg(at + bx^k \lfloor y \rfloor) - \deg t - \deg d \,.$$

Factoring Integers

ANTL contains a C++ implementation for factoring a composite integer $n \in \mathbb{Z}$ by Hirt based on the algorithms in Crandall and Pomerance [CP05, pp. 266, 344–345]. This algorithm works as follows. First we perform trial division on n to remove prime factors up to a bound of 1 000 000. If n is not completely factored by trial division, then Lenstra's elliptic curve method is used with an ECM stage-one bound of $B_E = 10\,000$ and stage-two bound of $100B_E$ [Len87]. Note that for the size of integers we were factoring (usually < 100 bits), using the number field sieve would not have provided an improvement.

Linear Algebra

ANTL relies on external libraries to perform linear algebra functions. Our goal was to use the LinBox library [Lin08] which contains efficient routines for sparse linear algebra. However, most of our attempts to use LinBox in our application failed and we were unable to get sufficient support from the developers to correct the problems. We resorted to using IML [CSF07] where LinBox failed. While this package does not provide sparse linear algebra, it does contain efficient routines for dense matrices. Consequently, due to the memory contraints of using dense matrices, we were unable to scale the inputs to our implementation as large as we would have liked.

One should note that getting a working implementation of index calculus in the infrastructure was our focus in this thesis. The fact that we were unable to get a sparse linear algebra implementation is unfortunate, but the linear algebra is not specific to our application. Our results in Section 5.4 separate the timings for each phase of index calculus. However, one should note that we did choose our parameters

to optimize the timings with the best linear algebra software we had available.

Ensuring Full Rank

We implemented the baby walk strategy described in Section 4.3.4 to compute a strictly diagonally dominant relation matrix. However, we discovered that this method is slower than just computing a baby walk starting from $\mathcal{O}$. Moreover, using large primes is not as effective when we are trying to get a diagonally dominant matrix. So we did the latter in practice, followed by a rank computation to ensure we have full rank.

With sieving and our baby walks in practice, we used `IML` to compute the row rank modulo a random small word-sized prime. This function also returns the row rank profile, *i.e.* a vector indicating which rows are linearly independent. Instead of naïvely generating more random relations, we used this row rank profile to generate relations that we know will be linearly independent. This prevents the relation matrix from getting as large as the naïve approach, thus improving the later linear algebra steps.

There is a small chance that the row rank and the profile will be incorrect given that we are not computing it over the integers. This will only happen if the random word-sized prime divides the determinant. However unlikely, if this does occur, we can detect that the rank is abnormally small and just choose another random prime.

5.3 Parameter Selection

To select our parameters for our results we relied on formulae provided by Velichka for imaginary quadratic function fields [Vel08, pp. 65–69, 84–88]. We briefly describe

those formulae here.

5.3.1 Smoothness Bound for Baby Walks

We wish to determine the best smoothness bound B to use when generating relations using the baby walk method. But first we must estimate the size of the factor base for a given smoothness bound B. The number of irreducible monic polynomials of degree n in $\mathbb{F}_q[x]$ is

$$I_n = \frac{1}{n}\sum_{d|n} \mu(d) q^{n/d} ,$$

where $\mu(\,\cdot\,)$ is the Möbius function [BS96, pp. 23, 134]. The Möbius function is defined as

$$\mu(d) = \begin{cases} 0 & \text{if } d \text{ is divisible by a square} \neq 1 \\ (-1)^t & \text{if } d = p_1 \cdots p_t, \text{ the product of } t \text{ distinct irreducible polynomials .} \end{cases}$$

Note that a prime ideal $p\mathcal{O}$ is ramified if and only if $p|f$ or $p|h$ (*cf.* Section 4.3.2). Thus we expect that the number of ramified prime ideals is negligible and we expect $\Phi(y) = y^2 + hy - f \pmod{p}$ to have a solution for approximately half the monic irreducible polynomials $p \in K[x]$. Then the number of ramified and splitting ideals of degree n to be put in the factor base is $A_n = \frac{1}{2} I_n$. Therefore, the size of the factor base will be $n_B \approx \sum_{n=1}^{B} A_n$.

Now we can optimize the smoothness bound B based on an estimate of how many baby steps are required to compute m relations. For our experiments in Section 5.4.2, we started with $m = n_B + 5$. We will first compute the number of baby steps in the

simpler case when no large primes are used. We use the following non-asymptotic estimate for the number of B-smooth reduced ideals extended from Jacobson, Menezes and Stein [JMS01, §5.1] by Maurer, Menezes and Teske [MMT02, §4.3]):

Theorem 5.8 (Maurer, Menezes & Teske (2002)). *For a given smoothness bound B, the number of B-smooth reduced ideals in the quadratic order $\mathcal{O}$ of a function field of genus g is*

$$n_{\mathcal{O}/B} = \sum_{i=1}^{g} \left[\prod_{n=1}^{B} \left(\frac{1+x^n}{1-x^n} \right)^{A_n} \right]_i ,$$

where $[\cdot]_i$ denotes the coefficient of x^i.

The value of $n_{\mathcal{O}/B}$ can be found by computing the first $g+1$ terms of the Taylor expansion of $\prod_{n=1}^{B} \left(\frac{1+x^n}{1-x^n}\right)^{A_n}$ about $x = 0$. Then, assuming Heuristic 4.11, we expect to find a B-smooth ideal every $E_B = \left\lceil \frac{h_{\mathcal{O}} R_{\mathcal{O}}}{n_{\mathcal{O}/B}} \right\rceil$ baby steps.

Since we initially find $n_B + 5$ relations before testing the relation matrix for full rank, we expect to find these relations in $T_B = (n_B+5)E_B$ baby steps. By computing T_B for $1 \leq B \leq g$, we choose the smoothness bound B with the minimum value of T_B.

Consider now the large prime variant of baby walks. Following Velichka's recommendations, we used a large prime bound of $B_L = B + 1$, implying that there are $n_{B_L} = 2A_{B+1}$ large primes ideals under the assumption that the number of ramified prime ideals is negligible. Since we only test $\mathcal{O}$-ideals of degree $\leq g$ (*i.e.* reduced) for (almost-)smoothness, it follows from Theorem 5.8 that the number of almost B-

smooth ideals is

$$n_{\mathcal{O}/B_L} = n_{B_L} \sum_{i=1}^{g-(B+1)} \left[\prod_{n=1}^{B} \left(\frac{1+x^n}{1-x^n} \right)^{A_n} \right]_i .$$

If we assume that Heuristic 4.11 also applies to almost B-smooth ideals, *i.e.* that they are evenly distributed between and within the ideal classes of $\mathrm{Cl}(\mathcal{O})$, then we expect to find a partial relation every $E'_{B_L} = \left\lceil \frac{h_{\mathcal{O}} R_{\mathcal{O}}}{n_{\mathcal{O}/B_L}} \right\rceil$ baby steps.

Now we need to estimate the number of full relations we expect to obtain from a set $\mathcal{P}_{B_L}$ of partial relations. Suppose at some point in time we have found $n_{\mathcal{P}}$ partial relations. The following result is from Thériault [Thé03, §5.6]:

Theorem 5.9 (Thériault, 2003). *Let $n_{\mathcal{R}}$ be the number of matching pairs found in a sample of size $n_{\mathcal{P}}$ chosen with replacement from n_{B_L} elements. If $3 \leq n_{\mathcal{P}} < \frac{1}{2} n_{B_L}$, then we have the bound*

$$\frac{2n_{\mathcal{P}}^2}{3n_{B_L}} \leq n_{\mathcal{R}} \leq \frac{n_{\mathcal{P}}^2}{n_{B_L}} .$$

Therefore, from Theorem 5.9 we expect to get at least one full relation once $n_{\mathcal{P}} = \sqrt{\frac{3}{2} n_{B_L}}$.

We have that for every almost B-smooth ideal found, the number of B-smooth ideals found is

$$\frac{E'_{B_L}}{E_B} = \left\lceil \frac{h_{\mathcal{O}} R_{\mathcal{O}}}{n_{\mathcal{O}/B_L}} \cdot \frac{n_{\mathcal{O}/B}}{h_{\mathcal{O}} R_{\mathcal{O}}} \right\rceil = \left\lceil \frac{n_{\mathcal{O}/B}}{n_{\mathcal{O}/B_L}} \right\rceil .$$

To compute the number of smooth and almost smooth ideal required to find $n_B + 5$

full relations, we solve for x in

$$\frac{2}{3n_{B_L}}x^2 + \left\lceil \frac{n_{\mathcal{O}/B}}{n_{\mathcal{O}/B_L}} \right\rceil x = n_B + 5 \, .$$

Then the number of baby steps required to find these x ideals is $T'_{B_L} = xE'_{B_L}$. Again, we choose the smoothness bound B such that T'_{B_L} is the minimum for $1 \leq B \leq g$ and $B_L = B + 1$.

For a particular bound B', the size of the factor base n_B increases dramatically from $B = B'$ to $B = B' + 1$ (especially for larger q). Therefore, in our implementation we allowed for fractional smoothness bounds. For example, we would allow a smoothness bound of $B = 3\frac{1}{4}$ that includes all the splitting and ramified prime ideals of at most degree 3, plus $\frac{1}{4}$ of the prime ideals of degree 4. This was done by only adding $\frac{1}{4}A_4 = \frac{1}{8}I_4$ prime ideals of degree 4 to the factor base.

For generating our numerical results, we chose the smoothness bound for a particular field size q and genus g as follows. Based on tests, we found an estimate $N_{\mathcal{F}}$ of the maximum n_B such that our linear algebra routines could handle the relation matrix in memory. Once we computed B to minimize T'_{B_L}, then we chose the largest smoothness bound $B' = kB \leq N_{\mathcal{F}}$ for $0 < k \leq 1$.

5.3.2 Sieve Parameters

For relation generation using sieving, we typically used the same smoothness bound chosen for the baby walk. However, there were occasions when the field size and genus were sufficiently small that increasing the smoothness bound gave a better result for sieving.

Other parameters for sieving include the sieve bound S, sieving interval M, tolerance value T, and the number of primes n_Q used to generate the ideal for the sieving polynomial in self-initialization. We did attempt to tweak these parameters a little from the suggestions made by Velichka, but we expect that the sieving runtime could be improved in many cases through careful adjustments.

First, we followed Velichka's suggestion to use a sieving interval of $M = B - 2$. Velichka claimed this choice was based on empirical evidence. Secondly, we wanted the sieve polynomial $g_\ell(u)$ to have a leading coefficient of degree $g - M$. Therefore, we attempted to generate the ideal $\mathfrak{a}_i$ such that it had degree $g - M$. Since n_Q is the number of factors of $\mathfrak{a}_i$, we needed each factor to have degree $(g - M)/n_Q \leq B$. Specifically, we used n_Q such that $(g - M)/n_Q \approx B - 1$.

The sieving tolerance T affects how many candidates will be selected in the sieving interval to be tested for smoothness. Velichka showed that smooth ideals with linear factors will be selected as candidates when $T = g + M$. Smooth ideals with square factors will be chosen as candidates if $T = g$. To select candidates with large prime factors for a bound $B_L = B + 1$, T should be reduced by $B + 1$. Following Velichka's recommendation, we used $T = \max\{g - M + 1 - 2B,\ 1\}$ as our starting tolerance value.

We started with the sieve bound $S = B$. In situations where the optimal smoothness bound B increased when increasing the genus g, we tried using a sieve bound of $S = B - 1$ as long as $\#\mathcal{F}_S$ was not too small (*e.g.* < 100). In these situations, we lowered the tolerance T so that we would still find candidates with factors of degree B.

5.4 Computations

In this section we present computational timings for computing the regulator using our implementation described previously. First we briefly mention the previous best results in the literature for computing the regulator.

5.4.1 Previous Results

We are aware of two sources in the literature that have provided timings for computing the regulator in real quadratic function fields. In 1998, Stein and Williams used the two-phase baby-step giant-step algorithm to compute regulators (*cf.* Section 3.2.5). They were able to compute an 80-bit regulator of a genus-3 function field over $\mathbb{F}_q$ with $q = 10000007$ using a 200Mhz Pentium Pro in 10 hours [SW98, §6]. In 2002, Stein and Teske used a parallized Pollard's rho method to compute a 94-bit regulator of a genus-3 function field with $q = 2155000013$. They estimated that on a single Sun Ultra Enterprise 450 their computations would have taken 55 days and 6 hours [ST02b, §4.2].

Due to the older machines that these previous results were computed on, it is difficult to compare these to our results. Moreover, their techniques adapt better to small genus function fields, whereas our index calculus algorithm applies to function fields where the genus is large in comparison to the field size q. We provide these previous results merely as an estimate of the computational feasibility of computing regulators in the past to show that we can now compute larger regulators (with updated hardware).

5.4.2 Current Results

We gathered our experimental results on Intel Pentium 4 dual processor machines with a 3.0 GHz clock speed, 2048 KB cache, and 1.0 GB of memory. All of our timings are presented in the form :ss, mm:ss, or hh:mm:ss (hours, minutes, and seconds).

We implemented both the determinant and kernel methods for computing the regulator (*cf.* Section 5.1.5). However, using `IML` for linear algebra, the determinant method always proved to be slower in practice, so we only present the kernel method timings in the tables below.

For relation generation we tested both baby walks and self-initialized sieving. All of the results are using large primes and partial relations. We started by generating $n_B + 5$ relations, then computed the rank profile modulo a small prime to generate relations specifically for the linearly dependent rows.

For the odd characteric fields we chose the field size q to be the largest prime less than a power of 2. The one exception to this rule was $\approx 2^4$ where we used 17 since it is closer to 16 than 13. Therefore, values of q used for odd characteristic fields were $3, 7, 17, 31, 61, 127, 251, 509, 1021$.

Tables 5.10 and 5.11 present our best times between baby walk and sieving for various genera and field sizes. Timings for both baby walks and sieving are given in Tables 5.14 and 5.15, split up for each step of index calculus. The parameters used to achieve these timings are given in Tables 5.12 and 5.13. We compare the total time for each relation generation method to an unoptimized baby-step giant-step implementation by Jacobson in Tables 5.16 and 5.17.

Table 5.10. *Timings for regulator computation, varying the genus g and field size q for even characteristic fields*

	g							
q	5	10	15	20	25	30	35	40
2^2	:00	:00	:02	:25	1:39	6:34	32:35	7:11:32
2^3	:00	:02	:21	5:50	57:43	8:52:58	—	—
2^4	:01	:03	1:43	5:28:04	—	—	—	—
2^5	:01	:32	58:09	—	—	—	—	—
2^6	:02	2:24	17:11:07	—	—	—	—	—
2^7	:10	35:54	—	—	—	—	—	—
2^8	:24	4:45:34	—	—	—	—	—	—
2^9	1:02	—	—	—	—	—	—	—
2^{10}	3:37	—	—	—	—	—	—	—

Table 5.11. *Timings for regulator computation, varying the genus g and field size q for odd characteristic fields*

	g							
q	5	10	15	20	25	30	35	40
$\approx 2^2$	:00	:00	:03	:01	:07	:31	4:22	22:00
$\approx 2^3$	:00	:01	:07	6:37	28:56	7:16:03	—	—
$\approx 2^4$	:00	:04	3:08	5:07:17	—	—	—	—
$\approx 2^5$	:03	:13	39:34	—	—	—	—	—
$\approx 2^6$	:03	2:14	13:31:10	—	—	—	—	—
$\approx 2^7$	:04	33:26	—	—	—	—	—	—
$\approx 2^8$	:13	5:27:40	—	—	—	—	—	—
$\approx 2^9$	:38	—	—	—	—	—	—	—
$\approx 2^{10}$	2:51	—	—	—	—	—	—	—

The column headings in Tables 5.12 and 5.13 use the following notation. The sieve bound is denoted by S; the sieving interval by M; the tolerance value is T; and n_Q is the number of prime ideals used to generate the sieving polynomial. These parameters were initially chosen according to the formulae in Section 5.3, then some of them have been tweaked to obtain faster runtimes or to satisfy the constraints of our linear algebra. The $\#\mathcal{R}_B$ column is not really a parameter that was set beforehand, but it gives the number of relations that we generated to obtain a full rank relation matrix. Note that in all cases we started by generating $\#\mathcal{F}_B + 5$ relations (except when the factor base was very small).

Table 5.12. *Relation generation parameters in even characteristic fields*

		Baby Walk			Sieving						
q	g	B	$\#\mathcal{F}_B$	$\#\mathcal{R}_B$	B	S	M	T	n_Q	$\#\mathcal{F}_B$	$\#\mathcal{R}_B$
2^2	5	2	4	7	3	3	1	2	2	15	20
2^2	10	3	15	25	3	3	1	4	3	15	20
2^2	15	4	42	48	4	4	2	6	4	42	47
2^2	20	5	129	139	5	5	3	8	4	129	135
2^2	25	6	466	507	6	6	4	10	4	466	483
2^2	30	6	459	485	6	6	4	15	5	459	470
2^2	35	7	1675	1795	7	7	5	17	5	1675	1730
2^2	40	$7\frac{1}{4}$	2672	2939	$7\frac{1}{4}$	$7\frac{1}{4}$	5	22	5	2672	2836
2^3	5	2	13	14	2	2	1	1	2	18	24
2^3	10	3	104	121	3	3	1	4	3	104	115
2^3	15	3	102	107	3	3	2	8	5	102	107
2^3	20	4	594	621	4	4	2	11	5	594	604
2^3	25	$4\frac{3}{4}$	3063	3361	$4\frac{4}{5}$	$4\frac{4}{5}$	3	12	6	3226	3363
2^3	30	5	3887	3991	5	5	3	13	6	3882	3933

Relation generation parameters in even characteristic fields (cont.)

		Baby Walk			Sieving						
q	g	B	$\#\mathcal{F}_B$	$\#\mathcal{R}_B$	B	S	M	T	n_Q	$\#\mathcal{F}_B$	$\#\mathcal{R}_B$
2^4	5	1	12	32	1	1	1	3	4	12	17
2^4	10	2	76	84	2	2	2	5	4	76	81
2^4	15	3	755	779	3	3	2	8	5	755	762
2^4	20	$3\frac{1}{4}$	2788	3157	$3\frac{1}{5}$	3	2	8	6	2380	2893
2^5	5	1	14	19	2	2	1	1	2	264	294
2^5	10	2	261	271	2	2	2	6	4	261	267
2^5	15	$2\frac{1}{2}$	2992	3152	$2\frac{1}{2}$	2	1	8	7	2992	3292
2^6	5	1	33	39	1	1	1	5	4	33	38
2^6	10	2	1019	1042	2	2	2	5	4	1019	1026
2^6	15	$2\frac{1}{50}$	1913	2384	$2\frac{1}{100}$	2	2	5	7	1476	1771
2^7	5	1	64	69	1	1	1	3	4	64	69
2^7	10	2	4152	4236	2	2	1	6	5	4152	4173
2^8	5	1	122	128	1	1	1	5	4	122	127
2^8	10	$1\frac{1}{5}$	3391	3549	$1\frac{3}{20}$	1	1	5	9	2575	2734
2^9	5	1	250	256	1	1	1	3	4	250	257
2^{10}	5	1	505	513	1	1	1	3	4	505	515

Table 5.13. *Relation generation parameters in odd characteristic fields*

		Baby Walk			Sieving						
q	g	B	$\#\mathcal{F}_B$	$\#\mathcal{R}_B$	B	S	M	T	n_Q	$\#\mathcal{F}_B$	$\#\mathcal{R}_B$
$\approx 2^2$	5	3	7	8	4	4	2	2	1	15	20
$\approx 2^2$	10	4	7	12	4	4	2	5	2	7	12
$\approx 2^2$	15	5	31	46	5	5	3	9	3	31	36
$\approx 2^2$	20	5	44	55	5	5	3	8	4	44	55

Relation generation parameters in odd characteristic fields (cont.)

		Baby Walk			Sieving						
q	g	B	$\#\mathcal{F}_B$	$\#\mathcal{R}_B$	B	S	M	T	n_Q	$\#\mathcal{F}_B$	$\#\mathcal{R}_B$
$\approx 2^2$	25	6	103	113	6	6	3	10	4	103	111
$\approx 2^2$	30	7	250	280	7	7	3	11	4	250	271
$\approx 2^2$	35	8	668	746	8	8	4	13	4	668	714
$\approx 2^2$	40	8	678	734	8	8	5	19	5	678	704
$\approx 2^3$	5	2	15	17	2	2	1	3	2	15	20
$\approx 2^3$	10	3	71	81	3	3	1	6	3	71	79
$\approx 2^3$	15	3	67	73	4	3	2	7	4	364	399
$\approx 2^3$	20	4	358	367	5	4	3	8	4	2044	2166
$\approx 2^3$	25	5	2067	2223	5	5	3	19	5	2067	2146
$\approx 2^3$	30	5	2044	2100	5	5	2	19	6	2044	2084
$\approx 2^4$	5	1	9	14	2	2	1	3	2	78	89
$\approx 2^4$	10	2	63	68	2	2	2	6	4	63	68
$\approx 2^4$	15	3	894	921	3	3	1	11	5	894	914
$\approx 2^4$	20	$3\frac{1}{10}$	1932	2277	$3\frac{1}{5}$	3	2	7	6	2973	3290
$\approx 2^5$	5	1	13	23	2	2	1	1	2	248	270
$\approx 2^5$	10	2	242	258	2	2	2	5	4	242	247
$\approx 2^5$	15	$2\frac{1}{2}$	2728	2941	$2\frac{1}{2}$	$2\frac{1}{2}$	1	13	7	2728	2829
$\approx 2^6$	5	1	30	40	1	1	1	5	4	30	35
$\approx 2^6$	10	2	973	995	2	2	1	8	5	973	980
$\approx 2^6$	15	$2\frac{3}{100}$	2080	2666	$2\frac{1}{50}$	2	1	5	7	1701	2063
$\approx 2^7$	5	1	69	74	1	1	1	5	4	69	74
$\approx 2^7$	10	2	4080	4209	2	2	1	8	5	4080	4121
$\approx 2^8$	5	1	117	122	1	1	1	5	4	117	122
$\approx 2^8$	10	$1\frac{1}{5}$	3262	3429	$1\frac{1}{5}$	1	1	5	9	3262	3301
$\approx 2^9$	5	1	262	270	1	1	1	5	4	262	267

Relation generation parameters in odd characteristic fields (cont.)

		Baby Walk			Sieving						
q	g	B	$\#\mathcal{F}_B$	$\#\mathcal{R}_B$	B	S	M	T	n_Q	$\#\mathcal{F}_B$	$\#\mathcal{R}_B$
$\approx 2^{10}$	5	1	496	505	1	1	1	5	4	496	502

The timings for each stage are given in Tables 5.14 and 5.15. The stages are factor base generation ($\mathcal{F}_B$ gen.), generation for the first $n_B + 5$ relations ($\mathcal{R}_B$ gen.), computing the rank and generating extra relations according to the rank profile (Rank), and finally computing a basis for the nullspace and computing the greatest common divisor of the distance of the prime ideals raised to the power of two random kernel vectors (Linalg.).

Table 5.14. *Timings for each stage of index calculus for computing the regulator in even characteristic fields*

		Baby Walk				Sieving			
q	g	$\mathcal{F}_B$ gen.	$\mathcal{R}_B$ gen.	Rank	Linalg.	$\mathcal{F}_B$ gen.	$\mathcal{R}_B$ gen.	Rank	Linalg.
2^2	5	:00	:00	:00	:00	:00	:00	:00	:00
2^2	10	:00	:00	:01	:00	:00	:00	:00	:00
2^2	15	:00	:02	:00	:00	:00	:02	:00	:00
2^2	20	:00	:28	:01	:00	:00	:23	:02	:00
2^2	25	:01	2:41	:12	:05	:01	1:18	:15	:05
2^2	30	:01	17:01	:50	:04	:01	5:29	:58	:06
2^2	35	:04	1:17:57	5:28	1:24	:04	21:42	9:37	1:12
2^2	40	:07	13:16:19	1:07:20	5:27	:07	4:30:14	2:34:17	6:54
2^3	5	:00	:00	:00	:00	:00	:00	:00	:00
2^3	10	:00	:02	:01	:00	:00	:02	:00	:00
2^3	15	:00	:23	:00	:00	:00	:21	:00	:00

Timings for each stage of index calculus for computing the regulator in even characteristic fields (cont.)

		Baby Walk				Sieving			
q	g	$\mathcal{F}_B$ gen.	$\mathcal{R}_B$ gen.	Rank	Linalg.	$\mathcal{F}_B$ gen.	$\mathcal{R}_B$ gen.	Rank	Linalg.
2^3	20	:01	6:42	:14	:06	:00	5:05	:39	:06
2^3	25	:03	1:36:36	10:52	7:11	:04	28:26	23:30	5:43
2^3	30	:05	27:33:19	39:19	9:58	:05	6:24:53	2:18:30	9:30
2^4	5	:00	:00	:02	:00	:00	:01	:00	:00
2^4	10	:00	:03	:00	:00	:00	:04	:00	:00
2^4	15	:00	3:18	:09	:09	:00	1:30	:04	:09
2^4	20	:03	4:47:54	32:57	7:10	:03	1:09:33	4:57:34	7:51
2^5	5	:00	:01	:00	:00	:00	:02	:02	:02
2^5	10	:00	:29	:01	:01	:00	:48	:01	:01
2^5	15	:02	1:29:42	4:52	4:51	:02	28:26	22:36	7:05
2^6	5	:00	:02	:00	:00	:00	:13	:00	:00
2^6	10	:00	3:05	:19	:15	:00	2:05	:05	:14
2^6	15	:01	15:48:51	2:46:21	4:36	:01	9:22:53	7:45:58	2:15
2^7	5	:00	:10	:00	:00	:00	:49	:00	:00
2^7	10	:02	30:08	5:32	10:33	:02	26:09	1:23	8:20
2^8	5	:00	:24	:00	:00	:00	19:26	:00	:00
2^8	10	:02	4:27:29	11:40	6:23	:01	4:37:29	2:08:18	3:53
2^9	5	:00	1:01	:00	:01	:00	11:23	:04	:01
2^{10}	5	:00	3:31	:03	:03	:00	49:55	:39	:03

Table 5.15. *Timings for each stage of index calculus for computing the regulator in odd characteristic fields*

		Baby Walk				Sieving			
q	g	$\mathcal{F}_B$ gen.	$\mathcal{R}_B$ gen.	Rank	Linalg.	$\mathcal{F}_B$ gen.	$\mathcal{R}_B$ gen.	Rank	Linalg.
$\approx 2^2$	5	:00	:00	:00	:00	:00	:00	:00	:00
$\approx 2^2$	10	:00	:01	:00	:00	:00	:00	:00	:00
$\approx 2^2$	15	:00	:01	:02	:00	:00	:02	:00	:01
$\approx 2^2$	20	:00	:01	:00	:00	:00	:09	:00	:00
$\approx 2^2$	25	:00	:06	:01	:00	:00	:27	:00	:00
$\approx 2^2$	30	:00	:27	:03	:01	:00	1:30	:02	:01
$\approx 2^2$	35	:00	3:46	:23	:13	:00	12:26	:11	:10
$\approx 2^2$	40	:00	20:15	1:34	:11	:00	1:42:22	1:05	:08
$\approx 2^3$	5	:00	:00	:00	:00	:00	:00	:00	:00
$\approx 2^3$	10	:00	:01	:01	:00	:00	:01	:00	:00
$\approx 2^3$	15	:00	:07	:00	:00	:00	:26	:01	:03
$\approx 2^3$	20	:00	6:31	:05	:01	:01	24:04	1:03	:01
$\approx 2^3$	25	:00	24:27	2:07	2:22	:00	2:03:18	1:49	1:47
$\approx 2^3$	30	:01	7:05:42	8:47	1:33	:01	45:15:29	33:16	1:57
$\approx 2^4$	5	:00	:01	:00	:00	:00	:00	:00	:00
$\approx 2^4$	10	:00	:04	:00	:00	:00	1:10	:00	:00
$\approx 2^4$	15	:00	2:46	:08	:14	:00	19:50	:05	:13
$\approx 2^4$	20	:01	4:24:45	39:00	3:31	:02	34:02:23	2:28:17	7:14
$\approx 2^5$	5	:00	:01	:01	:02	:00	:02	:00	:01
$\approx 2^5$	10	:00	:10	:02	:01	:00	1:51	:00	:01
$\approx 2^5$	15	:00	29:09	5:32	4:53	:00	4:27:48	13:42	3:32
$\approx 2^6$	5	:00	:02	:01	:00	:00	:10	:00	:00
$\approx 2^6$	10	:00	1:50	:10	:14	:00	16:54	:02	:11
$\approx 2^6$	15	:00	10:29:50	2:55:04	6:16	:00	165:58:39	1:33:40	3:08

Timings for each stage of index calculus for computing the regulator in odd characteristic fields (cont.)

		Baby Walk				Sieving			
q	g	$\mathcal{F}_B$ gen.	$\mathcal{R}_B$ gen.	Rank	Linalg.	$\mathcal{F}_B$ gen.	$\mathcal{R}_B$ gen.	Rank	Linalg.
$\approx 2^7$	5	:00	:04	:00	:00	:00	:16	:00	:00
$\approx 2^7$	10	:01	17:52	7:15	8:58	:01	1:01:15	:56	7:49
$\approx 2^8$	5	:00	:13	:00	:00	:00	1:22	:00	:00
$\approx 2^8$	10	:01	4:47:25	34:12	6:02	:01	177:28:16	4:44:01	4:33
$\approx 2^9$	5	:00	:35	:02	:01	:00	2:14	:00	:01
$\approx 2^{10}$	5	:00	2:45	:03	:03	:00	8:42	:01	:02

The total times for baby walk and sieving are given in Tables 5.16 and 5.17. We also compare to using the baby step giant step method to compute the regulator. We stress that the BSGS implementation used was not optimized and does not use any of the improvements described in Section 3.2.5.

Table 5.16. *Comparison of our index calculus methods to baby step giant step for computing the regulator in even characteristic fields*

q	g	$\log R_{\mathcal{O}}$	BSGS	Baby Walk	Sieving
2^2	5	9	:00	:00	:00
2^2	10	21	:00	:01	:00
2^2	15	31	:06	:02	:02
2^2	20	39	2:27	:29	:25
2^2	25	50	35:47:10	2:59	1:39
2^2	30	61	—	17:56	6:34
2^2	35	71	—	1:24:53	32:35
2^2	40	80	—	14:29:13	7:11:32

Comparison of our index calculus methods to baby step giant step for computing the regulator in even characteristic fields (cont.)

q	g	$\log R_{\mathcal{O}}$	BSGS	Baby Walk	Sieving
2^3	5	15	:00	:16	:00
2^3	10	31	:05	:03	:02
2^3	15	42	8:29	:23	:21
2^3	20	61	—	7:02	5:50
2^3	25	74	—	1:54:42	57:43
2^3	30	88	—	28:22:41	8:52:58
2^4	5	21	:00	:02	:01
2^4	10	41	9:20	:03	:04
2^4	15	56	—	3:36	1:43
2^4	20	80	—	5:28:04	6:15:01
2^5	5	24	:00	:01	:06
2^5	10	50	14:07:52	:31	:50
2^5	15	74	—	1:39:27	58:09
2^6	5	31	:03	:02	:13
2^6	10	61	—	3:39	2:24
2^6	15	91	—	18:39:49	17:11:07
2^7	5	36	:19	:10	:49
2^7	10	70	—	46:15	35:54
2^8	5	40	2:05	:24	19:26
2^8	10	81	—	4:45:34	6:49:41
2^9	5	45	17:04	1:02	11:28
2^{10}	5	47	36:03	3:37	50:37

Table 5.17. *Comparison of our index calculus methods to baby step giant step for computing the regulator in odd characteristic fields*

q	g	$\log R_{\mathcal{O}}$	BSGS	Baby Walk	Sieving
$\approx 2^2$	5	8	:00	:00	:00
$\approx 2^2$	10	17	:00	:01	:00
$\approx 2^2$	15	20	:00	:03	:03
$\approx 2^2$	20	33	:04	:01	:09
$\approx 2^2$	25	41	2:41	:07	:27
$\approx 2^2$	30	51	18:02:51	:31	1:33
$\approx 2^2$	35	58	—	4:22	12:47
$\approx 2^2$	40	64	—	22:00	1:43:35
$\approx 2^3$	5	15	:00	:07	:00
$\approx 2^3$	10	29	:01	:03	:01
$\approx 2^3$	15	41	:54	:07	:30
$\approx 2^3$	20	54	—	6:37	25:08
$\approx 2^3$	25	69	—	28:56	2:06:54
$\approx 2^3$	30	85	—	7:16:03	45:50:43
$\approx 2^4$	5	21	:00	:01	:00
$\approx 2^4$	10	44	4:30	:04	1:10
$\approx 2^4$	15	62	—	3:08	20:08
$\approx 2^4$	20	81	—	5:07:17	36:37:56
$\approx 2^5$	5	24	:00	:04	:03
$\approx 2^5$	10	51	3:49:03	:13	1:52
$\approx 2^5$	15	75	—	39:34	4:45:02
$\approx 2^6$	5	30	:01	:03	:10
$\approx 2^6$	10	60	—	2:14	17:07
$\approx 2^6$	15	85	—	13:31:10	167:34:27
$\approx 2^7$	5	36	:05	:04	:16

Comparison of our index calculus methods to baby step giant step for computing the regulator in odd characteristic fields (cont.)

q	g	$\log R_{\mathcal{O}}$	BSGS	Baby Walk	Sieving
$\approx 2^7$	10	67	—	33:26	1:10:01
$\approx 2^8$	5	39	:14	:13	1:22
$\approx 2^8$	10	80	—	5:27:40	182:16:51
$\approx 2^9$	5	46	1:53:25	:38	2:15
$\approx 2^{10}$	5	50	1:24:56	2:51	8:45

One will notice from the previous tables that in even characteristic fields sieving appears to be faster; however, in odd characteristic fields the baby walk strategy is significantly faster. Comparing the sieving results from Table 5.16 to Table 5.17 shows that sieving is still often faster in the odd characteristic case; this was expected since the odd characteristic fields are generally chosen to be smaller. However, the baby walk strategy is dramatically faster in the odd characteristic case versus the even case. Since the baby walk code is mostly templated, our hypothesis for this difference is the internal representation of field elements used by `NTL`. In the odd characteristic case we used the type `zz_pX` which is represented in a standard 32-bit integer, whereas in the even characteristic case we used the type `GF2EX` which is represented as a vector of binary polynomials `GF2X`, each also represented by a vector. This extra overhead in the even characteristic case could account for the dramatic difference in the baby walk strategy between even and odd characteristics. The internal representation may not affect sieving as much, since sieving is much more complicated and has its own overhead.

Due to our requirement to represent the matrices in dense format, memory was

the biggest issue in generating the previous results. For the larger genera we often had to choose the smoothness bound to be non-optimal in order to be able to handle the resulting matrices in memory. We ran a slightly larger example on an Intel Xeon 3.6 GHz machine with 1024KB cache and 6.2 GB of memory. These results are given in Table 5.18.

Table 5.18. *A larger regulator computation example*

q	g	$\log R_{\mathcal{O}}$	Strategy	$\#\mathcal{F}_B$	$\mathcal{F}_B$ gen.	$\mathcal{R}_B$ gen.	Rank	Linalg.	Total
$\approx 2^5$	20	99	Baby Walk	5165	:01	34:24:52	13:13	14:05	34:52:11

Chapter 6

Conclusions

For high genus real quadratic function fields, index calculus in the infrastructure provides a practical method for computing invariants like the regulator, class number and class group structure, as well as solving instances of the infrastructure DLP. For our "baby walk" relation generation method we provided a heuristic analysis, summarized in Table 6.1.

Table 6.1. *Heuristic, expected asymptotic runtime complexity for index calculus in the infrastructure*

Computing the regulator	$O(\mathbf{L}_{q^g}(2.83 + o(1)))$
Computing the class number and group structure	$O(\mathbf{L}_{q^g}(3.45 + o(1)))$
Solving an instance of the infrastructure DLP	$O(\mathbf{L}_{q^g}(2.45 + o(1)))$

We provided a description of our implementation of index calculus, as well as numerical results for computing the regulator. Our implementation was much faster than a standard, unoptimized implementation of baby-step giant-step. We were able

to compute a 99-bit regulator in a day and a half, whereas the best previous published result was a 91-bit regulator in 55 days.

We compared our baby walk relation generation method against a self-initialized sieve. Our results showed that in even characteristic fields sieving often outperformed the baby walk, but in odd characteristic fields the baby walk was dramatically faster.

6.1 Future Work

In this section we outline some areas for future work to improve or extend the contents of this thesis.

Sparse Linear Algebra

The numerical results in this thesis have been held back due to the lack of an implementation of efficient sparse linear algebra. This did not only affect the linear algebra step, but in many cases the timings for relation generation were increased due to the need to use lower smoothness bounds to allow the linear algebra to complete.

Implementing this sparse linear algebra was out of the scope of this thesis. We look forward to progress in the `LinBox` project [Lin08] that would allow the use of a blackbox library supporting the sparse linear algebra required in this work.

Sieve Parameters

We did not spend much time attempting to optimize the sieving parameters for index calculus in the infrastructure. We used the parameters as given by Velichka for imaginary quadratic function fields [Vel08, pp. 84–88]. Note that in real quadratic function fields one may be able to minimize the degree of the sieve polynomial to $g - M + 1$

rather than $g - M$ as was used for our computations. This would affect the number of factors n_Q of the sieving ideal as well as the tolerance value T. Consequently, we expect that a more thorough investigation would lead to faster sieving runtimes.

Multi-Large Primes

Allowing relations with multiple large primes has been used successfully in factoring algorithms (*cf.* Lenstra and Lenstra [LL93] and Dodson and Lenstra [DL95]). Blake *et al.* described a method to use two large primes to compute discrete logarithms [BFHMV84]. Holt and Davenport discussed multi-large prime variants for discrete logarithm computations and gave results for an implementation in the group $(\mathbb{Z}/p\mathbb{Z})^*$ for a prime p [HD03, Hol03]. An implementation and analysis of a double large prime variant of index calculus for low genus imaginary quadratic function fields was given by Gaudry, Thomé, Thériault and Diem [GTTD07]. It would be interesting to see how a multi-large prime variant could be extended to high genus real quadratic function fields.

Parallelization

Relation generation using sieving can be easily parallelized. Parallelizing the baby walks would be possible too if one was careful that the walks did not overlap. We did not experiment with parallelization, but it should be possible to obtain a speed-up for relation generation roughly inversely proportional to the number of computers used.

Theoretically, block variants of the sparse linear algebra would allow the linear algebra step to also be parallelized. We are not aware of any implementations.

Sieving in Low Genus

Gaudry described how to perform index calculus in low genus imaginary quadratic function fields using random walks [Gau00b]. This random walk method was improved by Thériault [Thé03] and by Gaudry, Thomé, Thériault and Diem [GTTD07]. We would like to see this low genus method extended to real quadratic function fields and investigate how sieving would perform in this situation.

Explicit Formulae

The runtime of the baby walk method for generating relations could be significantly improved by implementing explicit formulae for the baby step as discussed in Section 3.3.2.

Rigourous Complexity Analysis

Our complexity results for index calculus in the infrastructure are based on heuristic assumptions. Müller, Stein and Thiel had a rigourous analysis for their relation generation method [MST99, §4]. We are unaware of how to make our analysis rigourous when using the baby walk.

Analysis of Sieving and Large Primes

Thériault provided a complexity analysis of a large prime variant of index calculus in small genus imaginary quadratic function fields [Thé03, §5]. We were unsuccessful at obtaining a complexity for the large prime variant in large genus real quadratic function fields that showed an improvement over not using large primes. Due to the limited time available for this work, more analysis could prove successful. As far as

we know, this problem is also open for the large genus strategies of index calculus in imaginary quadratic function fields.

A complexity analysis of sieving in function fields seems to be more difficult.

Bibliography

[Abe94] Christine S. Abel. *Ein Algorithmus zur Berechnung der Klassenzahl und des Regulators reellquadratischer Ordnungen.* PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1994. Cited on page 119.

[Ach06] Jeffrey D. Achter. The distribution of class groups of function fields. *Journal of Pure and Applied Algebra*, 204(2):316–333, 2006. MR 2006h:11132. Cited on page 56.

[ADH94] Leonard M. Adleman, Jonathan DeMarrais, and Ming-Deh Huang. A subexponential algorithm for discrete logarithms over the rational subgroup of the Jacobians of large genus hyperelliptic curves over finite fields. In Leonard M. Adleman and Ming-Deh Huang, editors, *Algorithmic Number Theory—ANTS I (Ithaca, NY)*, volume 877 of *Lecture Notes in Computer Science*, pages 28–40. Springer-Verlag, 1994. MR 96b:11078. Cited on page 6.

[AHU74] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974. MR 54:1706. Cited on pages 52 and 63.

[AP95] William R. Alford and Carl Pomerance. Implementing the self-initializing quadratic sieve on a distributed network. In Alf J. van der Poorten, Igor E. Shparlinskiĭ, and Horst G. Zimmer, editors, *Number-Theoretic and Algebraic Methods in Computer Science (Moscow, 1993)*, pages 163–174. World Scientific Publishing, River Edge, NJ, 1995. MR 96k:11152. Cited on page 137.

[Art21] Emil Artin. *Quadratische Körper im Gebiete der höheren Kongruenzen.* PhD thesis, Universität Leipzig, Germany, 1921. Reprinted in *Mathematische Zeitschrift*, 19:153–246, 1924. Cited on pages 46, 55, and 66.

[AT06] Roberto M. Avanzi and Nicolas Thériault. Index calculus. In Cohen and Frey [CF06], chapter 20, pages 495–509. MR 2007f:14020. Cited on page 143.

[Bau99] Mark L. Bauer. A subexponential algorithm for solving the discrete logarithm problem in the Jacobian of high genus hyperelliptic curves over arbitrary fields. Preprint available from `http://math.ucalgary.ca/~mbauer/papers.html`, 1999. Cited on page 6.

[Bau01] Mark L. Bauer. *Function Field Arithmetic and Related Algorithms.* PhD thesis, University of Illinois at Urbana-Champaign, 2001. Cited on page 6.

[BCI85] Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, editors. *Advances in Cryptology—EUROCRYPT '84 (Paris, France)*, volume 209 of *Lecture Notes in Computer Science.* Springer-Verlag, 1985. Cited on pages 199 and 200.

[BD91] Johannes A. Buchmann and Stephan Düllmann. A probabilistic class group and regulator algorithm and its implementation. In Attila Pethő, Michael E. Pohst, Hugh C. Williams, and Horst Günter Zimmer, editors, *Computational number theory (Debrecen, 1989)*, pages 53–72. Walter de Gruyter, Berlin, 1991. MR 92m:11150. Cited on page 5.

[BD92] Johannes A. Buchmann and Stephan Düllmann. Distributed class group computation. *Teubner-Texte Informatik,* Festschrift aus Anlass des sechzigsten Geburtstages von Herrn Prof. Dr. G. Hotz, 1:69–79, 1992. MR 93e:11153. Cited on page 5.

[Ben81] Michael Ben-Or. Probabilistic algorithms in finite fields. In *Proceedings of the 22nd Annual Symposium on Foundations of Computer Science—FOCS '81 (Nashville, TN)*, pages 394–398. IEEE Computer Society, 1981. Cited on page 98.

[BFHMV84] Ian F. Blake, Ryoh Fuji-Hara, Ronald C. Mullin, and Scott A. Vanstone. Computing logarithms in finite fields of characteristic two. *SIAM Journal on Algebraic and Discrete Methods*, 5(2):276–285, 1984. Cited on page 176.

[BL04] Christina Birkenhake and Herbert Lange. *Complex Abelian Varieties.* Number 302 in Grundlehren der mathematischen Wissenschaften. Springer-Verlag, 2nd edition, 2004. MR 2005c:14001. First edition published in 1992. MR 94j:14001. Cited on page 45.

[BLP93] Joe P. Buhler, Hendrik W. Lenstra, Jr., and Carl Pomerance. Factoring integers with the number field sieve. In Lenstra and Lenstra [LL93], pages 50–94. MR 96m:11116. Cited on page 116.

[Bri98] Keith Briggs. Quad-precision floating-point arithmetic (doubledouble). C library, no longer available, 1998. First released under the GNU Public Licence (GPL) in 1996. Cited on page 204.

[BS96] Eric Bach and Jeffrey Shallit. *Algorithmic Number Theory, Volume 1: Efficient Algorithms.* Foundations of Computing Series. MIT Press, 1996. MR 97e:11157. Cited on pages 58, 96, 98, 101, 102, 104, and 155.

[Buc90] Johannes A. Buchmann. A subexponential algorithm for the determination of class groups and regulators of algebraic number fields. In Catherine Goldstein, editor, *Séminaire de Théorie des Nombres (Paris, 1988–1989)*, number 91 in Progress in Mathematics, pages 27–41. Birkhäuser, Boston, 1990. MR 92g:11125. Cited on pages 5, 105, and 112.

[Buh98] Joe P. Buhler, editor. *Algorithmic Number Theory—ANTS III (Portland, OR)*, volume 1423 of *Lecture Notes in Computer Science.* Springer-Verlag, 1998. Cited on pages 201 and 206.

[BV07] Johannes A. Buchmann and Ulrich Vollmer. *Binary Quadratic Forms: An Algorithmic Approach.* Number 20 in Algorithms and Computation in Mathematics. Springer, 2007. Cited on page 106.

[BW88] Johannes A. Buchmann and Hugh C. Williams. A key-exchange system based on imaginary quadratic fields. *Journal of Cryptology*, 1(2):107–118, June 1988. MR 90g:11166. Cited on page 4.

[BW90] Johannes A. Buchmann and Hugh C. Williams. A key exchange system based on real quadratic fields. In Gilles Brassard, editor, *Advances in Cryptology—CRYPTO '89 (Santa Barbara, CA)*, volume 435 of *Lecture Notes in Computer Science*, pages 335–343. Springer-Verlag, 1990. MR 91f:94014. Cited on pages 5 and 86.

[Can87] David G. Cantor. Computing in the Jacobian of a hyperelliptic curve. *Mathematics of Computation,* special issue dedicated to Daniel Shanks on the occasion of his 70th birthday, 48(177):95–101, January 1987. MR 88f:11118. Cited on page 51.

[CDO93] Henri Cohen, Francisco Diaz y Diaz, and Michel Olivier. Calculs de nombres de classes et de régulateurs de corps quadratiques en temps sous-exponentiel. In Sinnou David, editor, *Séminaire de Théorie des Nombres (Paris, 1990–1991)*, number 108 in Progress in Mathematics, pages 35–46. Birkhäuser, Boston, 1993. MR 94m:11151. Cited on page 5.

[CDO97] Henri Cohen, Francisco Diaz y Diaz, and Michel Olivier. Subexponential algorithms for class and unit group computations. In John Cannon and Derek Holt (eds.), Proceedings the 1st MAGMA Conference (London, 1993), *Journal of Symbolic Computation*, 24(3–4):433–441, 1997. MR 98m:11138. Cited on page 5.

[CF06] Henri Cohen and Gerhard Frey, editors. *Handbook of Elliptic and Hyperelliptic Curve Cryptography.* Discrete Mathematics and its Applications. Chapman & Hall/CRC, 2006. MR 2007f:14020. Cited on pages 180, 186, and 189.

[Che51] Claude Chevalley. *Introduction to the Theory of Algebraic Functions of One Variable.* Number 7 in Mathematical Surveys. American Mathematical Society, 1951. MR 13:64a. Cited on page 36.

[CL84] Henri Cohen and Hendrik W. Lenstra, Jr. Heuristics on class groups of number fields. In Hendrik Jager, editor, *Number Theory (Noordwijkerhout, The Netherlands)*, number 1068 in Lecture Notes in Mathematics, pages 33–62. Springer-Verlag, 1984. MR 85j:11144. Cited on pages 4 and 56.

[Coc73] Clifford C. Cocks. A note on 'Non-secret encryption'. Research report, Communications-Electronics Security Group (CESG), Great Britain, November 1973. Classified until 1997. Cited on page 2.

[Coh93] Henri Cohen. *A Course in Computational Algebraic Number Theory.* Number 138 in Graduate Texts in Mathematics. Springer-Verlag, 1993. MR 94i:11105. Cited on pages 114 and 151.

[Coh96] Henri Cohen, editor. *Algorithmic Number Theory—ANTS II (Bordeaux, France)*, volume 1122 of *Lecture Notes in Computer Science.* Springer-Verlag, 1996. Cited on pages 200, 203, and 206.

[CP05] Richard Crandall and Carl Pomerance. *Prime Numbers: A Computational Perspective.* Springer-Verlag, 2nd edition, 2005. MR 2006a:11005. First edition published in 2001. MR 2002a:11007. Cited on page 153.

[CSF07] Zhuliang Chen, Arne Storjohann, and Cory Fletcher. Integer matrix library (IML). C library, version 1.0.2, available from `http://www.cs.uwaterloo.ca/~z4chen/iml.html`, September 2007. Source freely available. Uses [WP08] for linear algebra routines and [Gra07] for long integer arithmetic. First version released in 2004. Cited on page 153.

[CW87] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing—STOC '87 (New York, NY)*, pages 1–6. ACM Press, 1987. Extended in [CW90]. Cited on page 185.

[CW90] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, March 1990. Full version of [CW87]. Cited on pages 114 and 185.

[CZ81] David G. Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, 36(154):587–592, April 1981. MR 82e:12020. Cited on page 104.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976. MR 55:10141. Cited on page 1.

[Die08] Claus Diem. *On arithmetic and the discrete logarithm problem in class groups of curves.* Habilitationsscrift submitted to the Fakultät für Mathematik und Informatik der Universität Leipzig, 5 July 2008. Available from `http://www.math.uni-leipzig.de/~diem/`. Cited on page 3.

[DL95] Bruce Dodson and Arjen K. Lenstra. NFS with four large primes: An explosive experiment. In Don Coppersmith, editor, *Advances in Cryptology—CRYPTO '95 (Santa Barbara, CA)*, volume 963 of *Lecture Notes in Computer Science*, pages 372–385. Springer-Verlag, 1995. Cited on page 176.

[Doc06] Christophe Doche. Exponentiation. In Cohen and Frey [CF06], chapter 9, pages 145–168. MR 2007f:14020. Cited on page 79.

[Dül91] Stephan Düllmann. *Ein Algorithmus zur Bestimmung der Klassengruppe positiv definiter binärer quadratischer Formen.* PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1991. Cited on page 119.

[EG02] Andreas Enge and Pierrick Gaudry. A general framework for subexponential discrete logarithm algorithms. *Acta Arithmetica*, 102(1):83–103, 2002. MR 2002k:11225. Previously released as research report LIX/RR/00/04, Laboratoire d'Informatique, l'École Polytechnique, 2000. Cited on pages 89, 90, 91, and 124.

[EGG+06] Wayne Eberly, Mark W. Giesbrecht, Pascal Giorgi, Arne Storjohann, and Gilles Villard. Solving sparse rational linear systems. In Jean-Guillaume Dumas, editor, *Proceedings of the 2006 International Sym-*

posium on Symbolic and Algebraic Computation—ISSAC 2006 (Genoa, Italy), pages 63–70. ACM Press, 2006. Previously available as arXiv:cs/0603082v1 in March 2006. Cited on page 151.

[EGG+07] Wayne Eberly, Mark W. Giesbrecht, Pascal Giorgi, Arne Storjohann, and Gilles Villard. Faster inversion and other black box matrix computations using efficient block projections. In Christopher W. Brown, editor, *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation—ISSAC 2007 (Waterloo, ON)*, pages 143–150. ACM Press, 2007. Previously available as arXiv:cs/0701188v1 in January 2007. Cited on page 151.

[EJS+07] Stefan Erickson, Michael J. Jacobson, Jr., Ning Shang, Shuo Shen, and Andreas Stein. Explicit formulas for real hyperelliptic curves of genus 2 in affine representation. In Claude Carlet and Berk Sunar, editors, *Arithmetic of Finite Fields—WAIFI 2007 (Madrid, Spain)*, volume 4547 of *Lecture Notes in Computer Science*, pages 202–218. Springer-Verlag, 2007. Cited on page 76.

[ElG85] Taher ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84 (Santa Barbara, CA)*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer-Verlag, 1985. Cited on page 3.

[Ell70] James H. Ellis. The possibility of Non-Secret digital encryption. Research report, Communications-Electronics Security Group (CESG),

Great Britain, January 1970. Existence classified until 1997. Cited on page 2.

[Ell87] James H. Ellis. The story of non-secret encryption. Historical article commissioned by the Communications-Electronics Security Group (CESG), Great Britain, 1987. Classified until 1997. Cited on page 2.

[Eng00] Andreas Enge. *Hyperelliptic Cryptosystems: Efficiency and Subexponential Attacks*. PhD thesis, Universität Augsburg, Germany, 2000. Cited on page 6.

[Eng01] Andreas Enge. How to distinguish hyperelliptic curves in even characteristic. In Kazimierz Alster, Jerzy Urbanowicz, and Hugh C. Williams, editors, *Public-Key Cryptography and Computational Number Theory (Warsaw, Poland)*, pages 49–58. Walter de Gruyter, Berlin, 2001. MR 2002k:11095. Cited on page 30.

[Eng02] Andreas Enge. Computing discrete logarithms in high-genus hyperelliptic Jacobians in provably subexponential time. *Mathematics of Computation*, 71(238):729–742, April 2002. MR 2003b:68083. Previously released as research report CORR 99-04, Department of Combinatorics & Optimization, University of Waterloo, 1999. Cited on pages 6, 7, 99, 100, 111, 113, and 127.

[ES02] Andreas Enge and Andreas Stein. Smooth ideals in hyperelliptic function fields. *Mathematics of Computation*, 71(239):1219–1230, 2002. MR 2003d:11170. Previously released as research report CORR 2000-08,

Department of Combinatorics & Optimization, University of Waterloo, 2000. Cited on page 109.

[FL06] Gerhard Frey and Tanja Lange. Background on curves and Jacobians. In Cohen and Frey [CF06], chapter 4, pages 45–85. MR 2007f:14020. Cited on pages 14 and 24.

[FP99] Ralf Flassenberg and Sachar Paulus. Sieving in function fields. *Experimental Mathematics*, 8(4):339–349, 1999. MR 2000j:11179. Previously released as technical report TI-13/97, Cryptography and Computeralgebra group, Department of Computer Science, Technische Universität Darmstadt, 1997. Cited on pages 6, 130, 132, and 133.

[Ful89] William Fulton. *Algebraic Curves: An Introduction to Algebraic Geometry.* Addison-Wesley, 1989. MR 47:1807. Reprint of the original published by W.A. Benjamin, 1969. Cited on pages 12, 13, 14, 15, 16, 18, 19, 20, 21, and 24.

[FW89] Eduardo Friedman and Lawrence C. Washington. On the distribution of divisor class groups of curves over a finite field. In Jean-Marie de Koninck and Claude Levesque, editors, *Théorie des Nombres (Québec, PQ): Comptes rendus de la Conférence internationale de théorie des nombres tenue à l'Université Laval, 5–18 juillet 1987*, pages 227–239. Walter de Gruyter, Berlin, 1989. MR 91e:11138. Cited on page 56.

[Gau86] Carl Friedrich Gauss. *Disquisitiones arithmeticae.* Springer-Verlag, 1986. (trans.) Arthur A. Clarke. First edition published by Gerh. Fleischer, Lipsiae in 1801. Second edition published in Latin by Königliche

Gesellschaft der Wissenschaften in 1870. Reprint of the English translation of the second edition originally published by Yale University Press in 1966. Cited on pages 3 and 4.

[Gau00a] Pierrick Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In Preneel [Pre00], pages 19–34. Cited on page 6.

[Gau00b] Pierrick Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In Preneel [Pre00], pages 19–34. Cited on page 177.

[Gau07] Pierrick Gaudry. Index calculus for abelian varieties and the elliptic curve discrete logarithm problem. Preprint available from `http://www.loria.fr/~gaudry/`, 2007. To be published in the *Journal of Symbolic Computation*. An earlier preprint first appeared in 2004. Cited on page 3.

[GH00] Pierrick Gaudry and Robert Harley. Counting points on hyperelliptic curves over finite fields. In Stevenhagen and Bosma [SB00], pages 313–332. MR 2002f:11072. Cited on page 5.

[GHMM08] Steven D. Galbraith, Michael Harrison, and David J. Mireles Morales. Efficient hyperelliptic arithmetic using balanced representation for divisors. In van der Poorten and Stein [vdPS08], pages 342–356. Extended and corrected version available as Cryptology ePrint Archive Report 2008/265, June 2008. Cited on pages 47 and 48.

[Gie01] Mark W. Giesbrecht. Fast computation of the Smith form of a sparse integer matrix. *Computational Complexity*, 10(1):41–69, 2001. MR 2003d:15014. Cited on page 121.

[Gol85] Dorian Goldfeld. Gauss' class number problem for imaginary quadratic fields. *Bulletin (New Series) of the American Mathematical Society*, 13(1):23–37, 1985. Cited on page 4.

[Gra07] Torbjörn Granlund. GNU multiple precision arithmetic library (GMP). C/C++ library, version 4.2.2, available from `http://www.gmplib.org`, September 2007. Open source software contributed to by many authors and released under the GNU Lesser General Public License (LGPL). First version released in 1991. Cited on pages 152, 185, 191, 197, and 204.

[GRV08] Thierry Gautier, Jean-Louis Roch, and Gilles Villard. Givaro: C++ library for arithmetic and algebraic computations. C++ library, version 3.2.10, available from `http://www-lmc.imag.fr/CASYS/LOGICIELS/givaro/`, April 2008. Open source software contributed to by many authors and released under the GNU Public Licence (GPL). Uses [Gra07] for long integer arithmetic. First version released in 1994. Cited on page 197.

[GTTD07] Pierrick Gaudry, Emmanuel Thomé, Nicolas Thériault, and Claus Diem. A double large prime variation for small genus hyperelliptic index calculus. *Mathematics of Computation*, 76(257):475–492, 2007. MR 2007j:11174. Previously released as INRIA research report 5764,

Institut National de Recherche en Informatique et en Automatique, Lorraine, 2005, and as research report CORR 2004-29, Department of Combinatorics & Optimization, University of Waterloo, 2004. Cited on pages 6, 149, 176, and 177.

[Har77] Robin Hartshorne. *Algebraic Geometry.* Number 52 in Graduate Texts in Mathematics. Springer-Verlag, 1977. MR 57:3116. Cited on pages 12, 14, 16, 17, 18, 21, 22, and 45.

[Har07] David Harvey. Kedlaya's algorithm in larger characteristic. *International Mathematics Research Notices*, 2007(22), Article ID rnm095, 2007. Also released as arXiv:math/0610973v2 in August 2007. First version appeared in October 2006. Cited on page 6.

[Has36] Helmut Hasse. Zur Theorie der abstrakten elliptischen Funktionenkörper I, II, III. *Journal für die reine und angewandte Mathematik (Crelle's Journal)*, 175:55–62, 69–88, 193–208, 1936. Earlier version appeared in *Nachrichten der Gesellschaft der Wissenschaften zu Göttingen* 1:119–129, 1935. Cited on page 95.

[HD03] Andrew J. Holt and James H. Davenport. Resolving large prime(s) variants for discrete logarithm computation. In Kenneth G. Paterson, editor, *Cryptography and Coding 2003 (Cirencester, UK)*, volume 2898 of *Lecture Notes in Computer Science*, pages 207–222. Springer-Verlag, 2003. Cited on pages 148 and 176.

[HI98] Ming-Deh Huang and Doug Ierardi. Counting points on curves over finite fields. *Journal of Symbolic Computation*, 25(1):1–21, January 1998. MR 98i:11040. Cited on page 5.

[HM89] James L. Hafner and Kevin S. McCurley. A rigorous subexponential algorithm for computation of class groups. *Journal of the American Mathematical Society*, 2(4):837–850, October 1989. MR 91f:11090. Cited on pages 4, 105, and 122.

[HM91] James L. Hafner and Kevin S. McCurley. Asymptotically fast triangularization of matrices over rings. *SIAM Journal of Computing*, 20(6):1068–1083, December 1991. Cited on page 114.

[Hol03] Andrew J. Holt. *On Computing Discrete Logarithms: Large Prime(s) Variants.* PhD thesis, University of Bath, UK, 2003. Cited on page 176.

[HW79] G. H. Hardy and Edward M. Wright. *An Introduction to the Theory of Numbers.* Oxford University Press, 5th edition, 1979. First edition published 1938. Cited on pages 44 and 59.

[Jac99] Michael J. Jacobson, Jr. *Subexponential Class Group Computation in Quadratic Orders.* PhD thesis, Technischen Universität Darmstadt, Germany, 1999. Cited on pages 130, 134, 137, 142, and 152.

[JMS01] Michael J. Jacobson, Jr., Alfred J. Menezes, and Andreas Stein. Solving elliptic curve discrete logarithm problems using Weil descent. *Journal of the Ramanujan Mathematical Society*, 16(3):231–260, 2001. MR 2002h:14039. Previously released as research report CORR 2001-31,

Department of Combinatorics & Optimization, University of Waterloo, 2001. Cited on pages 6 and 156.

[JSS07a] Michael J. Jacobson, Jr., Renate Scheidler, and Andreas Stein. Cryptographic protocols on real hyperelliptic curves. *Advances in Mathematics of Computation*, 1(2):197–221, 2007. Extends work from "Faster Cryptographic Key Exchange on Hyperelliptic Curves," Yellow Series preprint 847, University of Calgary, 2005. Cited on pages 7, 27, 80, 81, and 83.

[JSS07b] Michael J. Jacobson, Jr., Renate Scheidler, and Andreas Stein. Fast arithmetic on hyperelliptic curves via continued fraction expansions. In Tony Shaska, William C. Huffman, David Joyner, and Vasyl Ustimenko, editors, *Advances in Coding Theory and Cryptology*, number 2 in Series on Coding Theory and Cryptology, pages 201–244. World Scientific Publishing, Hackensack, NJ, 2007. Cited on pages 27, 50, 62, 75, and 76.

[JSW06] Michael J. Jacobson, Jr., Renate Scheidler, and Hugh C. Williams. An improved real-quadratic-field-based key exchange procedure. *Journal of Cryptology*, 19(2):211–239, April 2006. MR 2006k:94089. Previously released as Yellow Series preprint 845, University of Calgary, 2005. Cited on page 5.

[JvdP02] Michael J. Jacobson, Jr. and Alfred J. van der Poorten. Computational aspects of NUCOMP. In John Cannon, Claus Fieker, and David Kohel, editors, *Algorithmic Number Theory—ANTS V (Sydney, Australia)*, volume 2369 of *Lecture Notes in Computer Science*, pages 120–133. Springer-Verlag, 2002. MR 2004m:11208. Cited on page 75.

[Kal93] Erich Kaltofen. Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems. In *Applied Algebra, Algebraic Algorithms and Error-correcting Codes (San Juan, PR, 1993)*, number 673 in Lecture Notes in Computer Science, pages 195–212. Springer-Verlag, 1993. MR 94k:11134. Extended in [Kal95]. Cited on page 195.

[Kal95] Erich Kaltofen. Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems. *Mathematics of Computation*, 64(210):777–806, April 1995. MR 95f:65094. Extension of [Kal93]. Cited on page 195.

[Ked01] Kiran S. Kedlaya. Counting points on hyperelliptic curves using Monsky-Washnitzer cohomology. *Journal of the Ramanujan Mathematical Society*, 16(4):323–338, 2001. MR 2002m:14019. Previously released as arXiv:math/0105031. Errata published in *Journal of the Ramanujan Mathematical Society*, 18(4):417–418, 2003. MR 2005c:14027. Cited on page 6.

[Ked04] Kiran S. Kedlaya. Computing zeta functions via p-adic cohomology. In Duncan Buell, Jonathan W. Sands, and David S. Dummit, editors, *Algorithmic Number Theory—ANTS VI (Burlington, VT)*, volume 3076 of *Lecture Notes in Computer Science*, pages 1–17. Springer-Verlag, 2004. MR 2006a:14033. Cited on page 6.

[Knu97] Donald E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms.* Addison-Wesley, 3rd edition, 1997.

83i:68003 (2nd ed.). First edition published 1969. 44:3531. Cited on page 98.

[Kob89] Neal Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1(3):139–150, 1989. MR 90k:11165. Cited on pages 6 and 51.

[Kra22] Maurice B. Kraitchik. *Théorie des nombres*, volume 1. Gauthier-Villars, 1922. Cited on page 88.

[Kra24] Maurice B. Kraitchik. *Recherches sur la théories des nombres.* Gauthier-Villars, 1924. Cited on page 88.

[KS91] Erich Kaltofen and B. David Saunders. On Wiedemann's method of solving sparse linear systems. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC-9) (New Orleans, LA)*, number 539 in Lecture Notes in Computer Science, pages 29–38. Springer-Verlag, 1991. Cited on page 125.

[Küc97] Wolfgang W. Küchlin, editor. *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation—ISSAC '97 (Kihei, Maui, HI).* ACM Press, 1997. Cited on page 207.

[Len87] Hendrik W. Lenstra, Jr. Factoring integers with elliptic curves. *Annals of Mathematics,* 2nd series, 126(2):649–673, 1987. MR 89g:11125. Cited on page 153.

[Len00] Arjen Lenstra. Long integer package (LIP). C library, version 1.1, available from `http://www.win.tue.nl/~klenstra`, 2000. Source freely available. First version released in 1989. Cited on page 204.

[Lin08] LinBox Team. Project LinBox: Exact computational linear algebra. C++ library, version 1.1.5, available from `http://www.linalg.org/`, April 2008. Open source software contributed to by many authors and released under the GNU Lesser General Public License (LGPL). Uses [Gra07], [Sho08], [WP08], and [GRV08]. First version released in 2002. Cited on pages 153 and 175.

[LL93] Arjen K. Lenstra and Hendrik W. Lenstra, Jr., editors. *The development of the Number Field Sieve.* Number 1554 in Lecture Notes in Mathematics. Springer-Verlag, 1993. MR 96m:11116. Cited on pages 176, 182, and 197.

[LLMP90] Arjen K. Lenstra, Hendrik W. Lenstra, Jr., Mark S. Manasse, and John M. Pollard. The number field sieve. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing—STOC '90 (Baltimore, MD)*, pages 564–572. ACM Press, 1990. Also available in [LL93, pp. 11–42]. Cited on page 116.

[MB75] Michael A. Morrison and John Brillhart. A method of factoring and the factorization of F_7. *Mathematics of Computation,* special issue dedicated to Derrick Henry Lehmer on the occasion of his 70th birthday, 29(129):183–205, 1975. MR 51:8017. Cited on page 142.

[McC89] Kevin S. McCurley. Cryptographic key distribution and computation in class groups. In Mollin [Mol89], pages 459–479. MR 92e:11149. Previously released as IBM Research Report RJ 6433 (62551), Almaden Research Center, 1988. Cited on page 4.

[McC90] Kevin S. McCurley. The discrete logarithm problem. In Carl Pomerance, editor, *Cryptology and Computational Number Theory (Boulder, CO)*, number 42 in Proceedings of Symposia in Applied Mathematics, pages 49–74. American Mathematical Society, 1990. MR 92d:11133. Cited on page 88.

[McE69] Robert J. McEliece. Factorization of polynomials over finite fields. *Mathematics of Computation*, 23(108):861–867, October 1969. 41:1694a,. Cited on page 97.

[Mer78] Ralph C. Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, April 1978. Cited on page 1.

[Mes00] Jean-François Mestre. Utilisation de l'AGM pour le calcul de $E(\mathbb{F}_{2^n})$ et courbes de genre 2. Lettre adressée à Gaudry et Harley, December 2000. Available from `http://www.math.jussieu.fr/~mestre/`. Cited on page 5.

[Mes02] Jean-François Mestre. Algorithmes pour compter des points de courbes en petite caractéristique et un petit genre. Notes of David Lubicz from a lecture given in the Séminair de Cryptographie de l'Université de Rennes, March 2002. Available from `http://www.math.jussieu.fr/~mestre/`. Cited on page 5.

[MM08] David J. Mireles Morales. An analysis of the infrastructure in real function fields. Preprint available as Cryptology ePrint Archive Report 2008/299, July 2008. Cited on page 56.

[MMT02] Markus Maurer, Alfred J. Menezes, and Edlyn Teske. Analysis of the GHS Weil descent attack on the ECDLP over characteristic two finite fields of composite degree. *LMS Journal of Computation and Mathematics*, 5:127–174, 2002. Cited on page 156.

[Mol89] Richard A. Mollin, editor. *Number Theory and Applications (Banff, AB)*, number 265 in NATO Advanced Science Institute Series C: Mathematical and Physical Sciences. Kluwer Academic Press, Dordrecht, 1989. Cited on pages 197 and 204.

[MST99] Volker Müller, Andreas Stein, and Christoph Thiel. Computing discrete logarithms in real quadratic congruence function fields of large genus. *Mathematics of Computation*, 68(226):807–822, April 1999. MR 99i:11119. Previously released on Citeseer at `http://citeseer.ist.psu.edu/muller97computing.html` in 1997. Cited on pages 7, 92, 95, 97, 99, 105, 111, 112, 115, 120, 122, 127, and 177.

[Neu99] Jürgen Neukirch. *Algebraic Number Theory.* Number 322 in Grundlehren der mathematischen Wissenschaften. Springer-Verlag, 1999. (trans.) Norbert Schappacher. MR 2000m:11104. Original published as *Algebraische Zahlentheorie* in 1992. Cited on pages 36, 53, and 54.

[Odl85] Andrew M. Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. In Beth et al. [BCI85], pages 224–314. MR 87g:11022. Cited on pages 88 and 142.

[Pau96] Sachar Paulus. An algorithm of subexponential type computing the class group of quadratic orders over pricipal ideal domains. In Cohen [Coh96], pages 247–262. MR 98c:11143. Cited on page 6.

[Pil90] Jonathan Pila. Frobenius maps of abelian varieties and finding roots of unity in finite fields. *Mathematics of Computation*, 55(192):745–763, October 1990. MR 91a:11071. Cited on page 5.

[Pol78] John M. Pollard. Monte Carlo methods for index computation mod p. *Mathematics of Computation*, 32(143):918–924, July 1978. MR 58:10684. Cited on pages 3 and 74.

[Pom85] Carl Pomerance. The quadratic sieve factoring algorithm. In Beth et al. [BCI85], pages 169–182. MR 87d:11098. Cited on page 137.

[Pom96] Carl Pomerance. A tale of two sieves. *Notices of the American Mathematical Society*, 43(12):1473–1485, December 1996. Cited on page 130.

[PR99] Sachar Paulus and Hans-Georg Rück. Real and imaginary quadratic representations of hyperelliptic function fields. *Mathematics of Computation*, 68(227):1233–1241, 1999. MR 99i:11107. Previously released as technical report TI-14/97, Cryptography and Computeralgebra group, Department of Computer Science, Technische Universität Darmstadt, 1997. Cited on pages 30, 40, and 41.

[Pre00] Bart Preneel, editor. *Advances in Cryptology—EUROCRYPT 2000 (Bruges, Belgium)*, volume 1807 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000. Cited on page 190.

[PS98] Sachar Paulus and Andreas Stein. Comparing real and imaginary arithmetics for divisor class groups of hyperelliptic curves. In Buhler [Buh98], pages 576–591. MR 2000i:11098. Cited on page 62.

[Rei60] George W. Reitwiesner. Binary arithmetic. In Franz L. Alt, editor, *Advances in Computers*, volume 1, pages 231–308. Academic Press, New York, 1960. Cited on page 78.

[Ros02] Michael Rosen. *Number Theory in Function Fields.* Number 210 in Graduate Texts in Mathematics. Springer-Verlag, 2002. MR 2003d:11171. Cited on page 46.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard A. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. Cited on page 2.

[Sat00] Takakazu Satoh. The canonical lift of an ordinary elliptic curve over a finite field and its point counting. *Journal of the Ramanujan Mathematical Society*, 15(4):247–270, 2000. MR 2001j:11049. Cited on page 5.

[SB00] Peter Stevenhagen and Wieb Bosma, editors. *Algorithmic Number Theory—ANTS IV (Leiden, Netherlands)*, volume 1838 of *Lecture Notes in Computer Science.* Springer-Verlag, 2000. Cited on pages 190 and 207.

[SBW91] Renate Scheidler, Johannes A. Buchmann, and Hugh C. Williams. Implementation of a key exchange protocol using real quadratic fields. In

Ivan B. Damgård, editor, *Advances in Cryptology—EUROCRYPT '90 (Aarhus, Denmark)*, volume 473 of *Lecture Notes in Computer Science*, pages 98–109. Springer-Verlag, 1991. Shorter version of [SBW94]. Cited on page 5.

[SBW94] Renate Scheidler, Johannes A. Buchmann, and Hugh C. Williams. A key-exchange protocol using real quadratic fields. *Journal of Cryptology*, 7(3):171–199, September 1994. MR 96e:94015. Results are from [Sch93, Part II]. Cited on page 202.

[Sch31] Friedrich Karl Schmidt. Analytische Zahlentheorie in Körpern der Charakteristik p. *Mathematische Zeitschrift*, 33:1–32, 1931. Cited on page 55.

[Sch85] René Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Mathematics of Computation*, 44(170):483–494, April 1985. MR 86e:11122. Cited on page 5.

[Sch93] Renate Scheidler. *Applications of Algebraic Number Theory to Cryptography.* PhD thesis, University of Manitoba, Winnipeg, Canada, 1993. Cited on pages 5 and 202.

[Sch95] René Schoof. Counting points on elliptic curves over finite fields. *Journal de Théorie des Nombres de Bordeaux*, 7(1):219–254, 1995. Les Dix-huitièmes Journées Arithmétiques (Bordeaux, 1993). MR 97i:11070. Cited on page 5.

[Sch96] Renate Scheidler. Compact representation in real quadratic congruence function fields. In Cohen [Coh96], pages 323–336. MR 98c:11126. Cited on pages 50, 55, and 57.

[Sem04] Igor Semaev. Summation polynomials and the discrete logarithm problem on elliptic curves. Preprint available from `http://eprint.iacr.org/2004/031`, 2004. Cited on page 3.

[Sey87] Martin Seysen. A probabilistic factorization algorithm with quadratic forms of negative discriminant. *Mathematics of Computation*, 48(178):757–780, April 1987. MR 88d:11129. Cited on pages 106 and 112.

[Sha71] Daniel Shanks. Class number, a theory of factorization and genera. In Donald J. Lewis, editor, *1969 Number Theory Institute (Stony Brook, NY)*, volume 20 of *Proceedings of Symposia in Pure Mathematics*, pages 415–440. American Mathematical Society, 1971. MR 47:4932. Cited on pages 4 and 72.

[Sha72a] Daniel Shanks. Five number-theoretic algorithms. In R.S.D. Thomas and Hugh C. Williams, editors, *Proceedings of the Second Manitoba Conference on Numerical Mathematics (Winnipeg, MB)*, number VII in Congressus Numerantium, pages 51–70. Utilitas Mathematica, Winnipeg, MB, 1972. MR 51:8072. Cited on page 96.

[Sha72b] Daniel Shanks. The infrastructure of a real quadratic field and its applications. In *Proceedings of the 1972 Number Theory Conference (Boulder, CO)*, pages 217–224, 1972. MR 52:10672. Cited on pages 4, 57, and 64.

[Sha89] Daniel Shanks. On Gauss and composition I, II. In Mollin [Mol89], pages 163–204. MR 92e:11150. Cited on page 75.

[Sho08] Victor Shoup. A library for doing number theory (NTL). C++ library, version 5.4.2, available from `http://www.shoup.net`, March 2008. Source freely available. Uses either [Len00] or [Gra07] for long integer arithmetic and includes parts of [Bri98] for floating-point arithmetic. First version released in 1990. Cited on pages 152 and 197.

[Sil86] Joseph H. Silverman. *The Arithmetic of Elliptic Curves.* Number 106 in Graduate Texts in Mathematics. Springer-Verlag, 1986. MR 87g:11070. Cited on page 23.

[Sil00] Joseph H. Silverman. The Xedni calculus and the elliptic curve discrete logarithm problem. *Designs, Codes and Cryptography*, 20:5–40, 2000. Cited on page 3.

[SL96] Arne Storjohann and George Labahn. Asymptotically fast computation of Hermite normal forms of integer matrices. In Yagati N. Lakshman, editor, *Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation—ISSAC '96 (Zürich, Switzerland)*, pages 259–266. ACM Press, 1996. Cited on page 114.

[Sma97] Nigel Smart. Experiments using an analogue of the number field sieve algorithm to solve the discrete logarithm problem in the Jacobians of hyperelliptic curves. Technical Report HPL-97-130, HP Laboratories, Bristol, UK, 1997. Cited on page 6.

[SSW96] Renate Scheidler, Andreas Stein, and Hugh C. Williams. Key-exchange in real quadratic congruence function fields. *Designs, Codes and Cryptography,* special issue dedicated to Dr. Gustavus J. Simmons, 7:153–174, 1996. MR 97d:94009. Cited on pages 7, 71, 83, and 86.

[ST02a] Andreas Stein and Edlyn Teske. Explicit bounds and heuristics on class numbers in hyperelliptic function fields. *Mathematics of Computation,* 71(238):837–861, April 2002. MR 2002k:11210. Cited on pages 74, 118, 119, and 120.

[ST02b] Andreas Stein and Edlyn Teske. The parallelized Pollard kangaroo method in real quadratic function fields. *Mathematics of Computation,* 71(238):793–814, April 2002. MR 2002k:11227. Previously released as research report CORR 2000-35, Department of Combinatorics & Optimization, University of Waterloo, 2000. Cited on pages 74 and 160.

[Ste99] Andreas Stein. Infrastructure in real quadratic function fields. Research Report CORR 99-17, Department of Combinatorics & Optimization, University of Waterloo, May 1999. Cited on page 53.

[Ste01] Andreas Stein. Sharp upper bounds for arithmetics in hyperelliptic function fields. *Journal of the Ramanujan Mathematical Society,* 16(2):1–86, 2001. MR 2002d:11134. Previously released as research report CORR 99-23, Department of Combinatorics & Optimization, University of Waterloo, 1999. Cited on pages 62 and 64.

[Sti93] Henning Stichtenoth. *Algebraic Function Fields and Codes.* Universitext. Springer-Verlag, 1993. MR 94k:14016. Cited on pages 24, 30, 31, 32, 33, 34, 35, 37, 38, 39, 40, 96, and 118.

[SW98] Andreas Stein and Hugh C. Williams. An improved method of computing the regulator of a real quadratic function field. In Buhler [Buh98], pages 607–620. MR 2000j:11201. Cited on pages 74, 115, and 160.

[SW99] Andreas Stein and Hugh C. Williams. Some methods for evaluating the regulator of a real quadratic function field. *Experimental Mathematics*, 8(2):119–133, 1999. MR 2000f:11152. Extends results of an unpublished manuscript entitled "Baby Step Giant Step in Real Quadratic Function Fields" that appeared in 1995. Cited on pages 49, 59, 64, 69, 72, and 74.

[SWD96] Oliver Schirokauer, Damian Weber, and Thomas Denny. Discrete logarithms: The effectiveness of the index calculus method. In Cohen [Coh96], pages 337–361. MR 98i:11109. Cited on page 88.

[Thé03] Nicolas Thériault. Index calculus attack for hyperelliptic curves of small genus. In Chi Sung Laih, editor, *Advances in Cryptology—ASIACRYPT 2003 (Taipei, Taiwan)*, volume 2894 of *Lecture Notes in Computer Science*, pages 75–92. Springer-Verlag, 2003. Cited on pages 6, 157, and 177.

[TS05] Edlyn Teske and Andreas Stein. Optimized baby step-giant step methods. *Journal of the Ramanujan Mathematical Society*, 20:1–32, 2005. MR 2005m:11238. Previously released as technical report CACR 2005-

11, Centre for Applied Cryptographic Research, University of Waterloo, 2005. Cited on pages 73 and 74.

[vdP03] Alfred J. van der Poorten. A note on NUCOMP. *Mathematics of Computation*, 72(244):1935–1946, October 2003. MR 2004b:11173. Cited on page 75.

[vdPS08] Alfred J. van der Poorten and Andreas Stein, editors. *Algorithmic Number Theory—ANTS VIII (Banff, AB)*, volume 5011 of *Lecture Notes in Computer Science*. Springer-Verlag, 2008. Cited on page 190.

[Vel08] Mark D. Velichka. Improvements to index calculus algorithms for solving the hyperelliptic curve discrete logarithm problem over characteristic two finite fields. Master's thesis, University of Calgary, Canada, 2008. Cited on pages 6, 101, 131, 132, 137, 138, 141, 142, 146, 152, 154, and 175.

[Vil97a] Gilles Villard. Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems. In Küchlin [Küc97], pages 32–39. Extended abstract of [Vil97b]. Cited on page 207.

[Vil97b] Gilles Villard. A study of Coppersmith's block Wiedemann algorithm using matrix polynomials. Research Report 975-I-M, IMAG Grenoble France, 1997. Full version of [Vil97a]. Cited on page 207.

[Vol00] Ulrich Vollmer. Asymptotically fast discrete logarithms in quadratic number fields. In Stevenhagen and Bosma [SB00], pages 581–594. MR 2003b:11135. Cited on page 4.

[vzGG03] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra.* Cambridge University Press, 2nd edition, 2003. MR 2004g:68202. First edition published in 1999. MR 2000j:68205. Cited on page 102.

[Wei48] André Weil. *Sur les courbes algébriques et les variétès qui s'en déduisent.* Number 1041 in Actualités scientifiques et industrielles. Hermann, Paris, 1948. MR 10:262c. Cited on page 95.

[Wie86] Douglas H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, IT-32(1):54–62, 1986. MR 87g:11166. Cited on pages 125 and 150.

[Wil74] Malcolm J. Williamson. Non-secret encryption using a finite field. Research report, Communications-Electronics Security Group (CESG), Great Britain, January 1974. Classified until 1997. Cited on page 2.

[WM68] Alfred E. Western and J. C. P. Miller. *Indices and Primitive Roots.* Number 9 in Royal Society Mathematical Tables. Cambridge University Press, 1968. MR 39:7792. Cited on page 88.

[WP08] R. Clint Whaley and Antoine Petitet. Automatically tuned linear algebra software (ATLAS). C library, version 3.8.1, available from `http://math-atlas.sourceforge.net`, February 2008. Open source software contributed to by many authors. First version released in 1997. Cited on pages 185 and 197.

[WW87] Hugh C. Williams and Marvin C. Wunderlich. On the parallel generation of the residues for the continued fraction factoring algorithm.

Mathematics of Computation, special issue dedicated to Daniel Shanks on the occasion of his 70th birthday, 48(177):405–442, January 1987. MR 88i:11099. Cited on pages 49, 59, and 60.

[Yun77] David Y. Y. Yun. Fast algorithm for rational function integration. In Bruce Gilchrist, editor, *Information Processing 77 (Toronto, ON)*, number 7 in IFIP Congress Series, pages 493–498. North-Holland, Amsterdam, 1977. Cited on page 104.

[ZS75] Oscar Zariski and Pierre Samuel. *Commutative Algebra,* volumes I & II. Number 28 & 29 in Graduate Texts in Mathematics. Springer-Verlag, 1975. Reprint of the originals published in the University Series in Higher Mathematics by D. Van Nostrand, Princeton in 1958–1960. MR 19:833e, MR 22:11006. Cited on pages 12, 15, 30, 31, 32, 33, 48, 49, 50, and 52.

[Zuc97a] Robert J. Zuccherato. The continued fraction algorithm and regulator for quadratic function fields of characteristic 2. *Journal of algebra,* 190(2):563–587, 1997. MR 98a:11156. Results also printed in [Zuc97b, Ch. 4–5]. Cited on pages 49, 59, 62, 64, 69, 72, and 73.

[Zuc97b] Robert J. Zuccherato. *New Applications of Elliptic Curves and Function Fields in Cryptography.* PhD thesis, University of Waterloo, Canada, 1997. Cited on page 209.