

THE UNIVERSITY OF CALGARY

Fast Ideal Arithmetic in Quadratic Fields

by

Reginald Sawilla

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
CROSS-DISCIPLINARY DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF MATHEMATICS AND STATISTICS

and

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

AUGUST, 2004

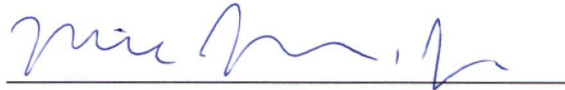
© Reginald Sawilla 2004

THE 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 "Fast Ideal Arithmetic in Quadratic Fields" submitted by Reginald Sawilla in partial fulfillment of the requirements for the degree of CROSS-DISCIPLINARY DEGREE OF MASTER OF SCIENCE.



Chair, Supervisor, Dr. H.C. Williams
Department of Mathematics and Statistics



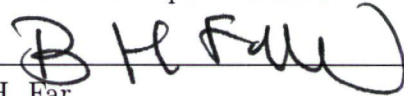
Co-Supervisor, Dr. M.J. Jacobson
Department of Computer Science



Dr. M.L. Bauer
Department of Mathematics and Statistics



Dr. R. Scheidler
Department of Computer Science



Dr. B.H. Far
Department of Electrical and Computer Engineering

June 28/04

Date

Abstract

Ideal multiplication and reduction are fundamental operations on ideals and are used extensively in class group and infrastructure computations; hence, the efficiency of these operations is extremely important. In this thesis we focus on reduction in real quadratic fields and examine all of the known reduction algorithms, converting them whenever required to work with ideals of positive discriminant. We begin with the classical algorithms of Gauss and Lagrange and move on to the algorithm Rickert developed for the closely related case of positive definite binary quadratic forms. Given any reduction technique, we present a general method computing the relative generator necessary for infrastructure computations. Rickert's algorithm along with an algorithm of Schönhage are adapted to ideals of real quadratic fields. We present a new method which combines the ideas of Lehmer, Williams, and others into a particularly simple algorithm. All of these algorithms have been implemented and compared with the Jacobson-Scheidler-Williams adaptation of NUCOMP in a cryptographic public key-exchange protocol. We conclude showing that Schönhage's algorithm is asymptotically the fastest but not useful in practice, JSW-NUCOMP is the fastest practical method when multiplying and reducing, and the new algorithm is the fastest method when only reduction is required.

Acknowledgments

I would like to thank my supervisor, Dr. Hugh Williams, for his invaluable advice, both professional and scholarly. His tireless work in establishing the Centre for Information Security and Cryptography (CISaC) has developed a fertile research environment among the best anywhere. This environment and the opportunity to study under his tutelage are the primary reasons I chose the University of Calgary.

I also thank my co-supervisor, Dr. Michael Jacobson. His friendly encouragement and patient help have been greatly appreciated on many occasions. The camaraderie we have shared has contributed significantly to the enjoyment of this experience. I thank both of my supervisors for their excellence and the huge role they have played in the development of this thesis.

Thank you to the other members of my examining committee, Dr. Mark Bauer, Dr. Renate Scheidler and Dr. Behrouz Far. The time and effort you have spent reading the thesis and giving advice is greatly appreciated. I would like to especially thank Dr. Scheidler who has put up with more of me than duty requires due to her close proximity to my office.

I have so appreciated my fellow students for their practical help with academic questions, patience in listening to my talks, and for their friendship. I have thoroughly

enjoyed the good times, food and games we have shared and have learned many things about myself, Calgary and other countries and cultures through them.

I would like to thank Dr. Amir Akbary at the University of Lethbridge for introducing me to number theory and cryptography. In two summer research sessions he generously gave me the freedom to pick any mathematical topic of study and pursue any angle of research that interested me. He liberally gave of his time and resources and was a significant factor in my decision to pursue graduate studies.

I am grateful to NSERC, iCORE, Dr. Williams, Dr. Scheidler and the University of Calgary for the funding provided to make this degree possible. The funding has taken many forms including research stipends, funds to attend conferences and workshops, and provision of computer equipment, all of which has enriched the academic experience.

I especially thank my wife Darcie who has been so supportive throughout my education. Similarly, I am indebted to each of my children Shafir, Aliya, Ciara, and Eve who have all made sacrifices.

Most of all I am grateful to God for his guidance, wisdom and favour. Apart from Him, I can do nothing.

Dedication

I dedicate this thesis to:

My father: Though his time on earth was cut short, his influence upon me has been profound.

My mother: To whom I am forever indebted for her endless encouragement, support and prayers.

My wife: The most important person in my life and best friend.

Contents

Approval Page	ii
Abstract	iii
Acknowledgments	iv
Dedication	vi
Contents	vii
List of Tables	ix
List of Figures	x
List of Algorithms	xi
Frequently Used Notation	xii
1 Introduction	1
1.1 Ideal Arithmetic	1
1.2 Organization Of The Thesis	4
2 Algebraic Number Theory Concepts	6
2.1 Algebraic Number Fields	6
2.2 Ideals	14
2.3 Class Group and Class Number	18
3 Multiplication and Reduction Theory	24
3.1 Multiplication	25
3.2 Equivalence	26
3.2.1 Equivalence Classes	26
3.2.2 Relative Generator	31

3.3	Reduction	34
3.3.1	Purpose of Reduction	34
3.3.2	Classical Reduction of Ideals (Lagrange)	34
3.4	Finding a Reduced Representative	41
3.5	Binary Quadratic Forms	42
4	Applications to Secure Communication	51
5	Survey of Improved Algorithms	57
5.1	Rickert's Algorithm	58
5.2	Schönhage's Algorithm	63
5.3	Schnorr and Seysen's Algorithm	73
5.4	A New Algorithm For Ideal Reduction	78
5.5	Shanks' Algorithm – NUCOMP	84
5.6	Improvements to NUCOMP	87
5.7	Complexity	92
6	Implementation and Timings	95
6.1	General Principles	95
6.2	Lehmer's Extended GCD Algorithm	98
6.3	Rickert's Algorithm	101
6.4	Schönhage's Algorithm	103
6.5	A New Algorithm For Ideal Reduction	107
6.6	Jacobson-Scheidler-Williams NUCOMP	109
6.7	Timings	112
7	Conclusion	119
	Bibliography	122
A	<i>An Improved Composition Algorithm</i>	129

List of Tables

5.1	Known Reduction Algorithm Complexity Results	94
6.1	Calculation of Q_i , P_i and R_i	109
6.2	Average Key Exchange Times In Seconds	114
6.3	Single Key Exchange Times In Seconds	115
6.4	Estimating Functions For Key Exchange Times	117

List of Figures

4.1	Message And Key Transmission	52
5.1	Magic Matrix	85
6.1	Key Exchange Comparison	115
6.2	Key Exchange Comparison (Logarithmic)	116

List of Algorithms

3.1	Rho(ρ) — one continued fraction expansion step	36
3.2	Continued Fraction Reduction (Matrix)	37
3.3	Continued Fraction Reduction (Ideal)	38
3.4	Continued Fraction Reduction (Implementation)	39
3.5	Near Reduced Ideal (Ideal)	42
3.6	Gaussian Reduction (Matrix)	48
3.7	Gaussian Reduction (Ideal)	49
3.8	Gaussian Reduction (Implementation)	50
5.1	Rickert–Style Reduction Algorithm (Matrix)	61
5.2	Rickert–Style Reduction Algorithm (Ideal)	62
5.3	Schönhage Reduction (Matrix)	64
5.4	Make Positive (Matrix)	65
5.5	Monotone Reduction — MR (Matrix)	66
5.6	Simple Step Above 2^m (Matrix)	71
5.7	Simple Step Above 2^m (Ideal)	72
5.8	Efficient Ideal Reduction (Ideal)	82
5.9	Efficient Ideal Reduction (Matrix)	83
5.10	Jacobson-Scheidler-Williams NUCOMP (Basic)	93
6.1	Lehmer’s Extended GCD (with Jebelean’s condition)	100
6.2	Rickert–Style Reduction Algorithm (Implementation)	102
6.3	Schönhage Reduction (Implementation)	103
6.4	Make Positive (Implementation)	104
6.5	Monotone Reduction — MR (Implementation)	105
6.6	Simple Step Above 2^m (Implementation)	106
6.7	Efficient Ideal Reduction (Implementation)	108
6.8	Jacobson-Scheidler-Williams NUCOMP (Implementation)	111

Frequently Used Notation

General

\approx	Approximately equal to	23
\in	Is an element of	7
\subseteq	Subset or subring	7
\sum	Summation	15
\prod	Product	61
$\lceil a \rceil$	Ceiling of a	61
$\lfloor a \rfloor$	Floor of a	23
\hat{a}	Single-precision approximation of a	58
$ a $	Absolute value of a	23
(a, b, c)	Binary quadratic form	43
$[q_0, \dots, q_{n-1}, \alpha_n]$	Simple continued fraction expansion	35
$\det M$	Determinant of the matrix M	29
$\begin{pmatrix} Q_i & P_i \\ P_i & R_i \end{pmatrix}$	A matrix	26
$\{a_i T\}$	Set comprised of elements a_i subject to condition T	15
$a \leftarrow b$	Set a to the value of b	36
A_i, B_i	Sequences from a simple continued fraction expansion	35
$c a$	c divides a	21

I_2	2×2 identity matrix.....	27
\mathbb{C}	Field of complex numbers.....	7
\mathbb{Q}	Field of rational numbers.....	7
\mathbb{Z}	Ring of rational integers.....	8
gcd	Greatest common divisor.....	25

Functions

λ_M	Function representing the action of the matrix M on a quadratic irrational.....	28
ϕ_N	Function representing the action of the matrix N on a 2×2 matrix.....	29
$f : A \rightarrow B$	A function f mapping from A into B	12
$a \mapsto b$	Element a maps to the element b	29
$f(\alpha)$	Polynomial f evaluated at α	7
$f(n) = \mathcal{O}(g(n))$	Big-O, $f(n) \leq cg(n)$ for some constant c when n is sufficiently large.....	34

Groups, Rings and Fields

$[E : F]$	Degree of E over F	7
$\bar{\alpha}$	Conjugate of the quadratic irrational element α	14
Δ_K	Discriminant of the number field K	12
\mathcal{O}_K	Ring of integers of the number field K	9
$\text{PGL}(2, \mathbb{Z})$	The factor group $\text{GL}(2, \mathbb{Z}) / \pm I_2$	27
$\mathbb{Q}(\sqrt{D})$	Quadratic number field and D is a square-free integer.....	12
$\{\alpha_1, \alpha_2, \dots, \alpha_n\}$	Set or basis.....	11
$F(\alpha)$	Extension field formed by adjoining α to F	7
K	An algebraic number field.....	8

Ideals

σ	$\sigma = 2$ if $D \equiv 1 \pmod{4}$, and $\sigma = 1$ otherwise	13
$\mathfrak{a}\mathfrak{b}$	Ideal \mathfrak{a} multiplied by ideal \mathfrak{b}	15
$\mathfrak{a} \sim \mathfrak{b}$	Ideal \mathfrak{a} is equivalent to ideal \mathfrak{b}	19
$[\mathfrak{a}]$	Equivalence class of the ideal \mathfrak{a}	19
$a\mathbb{Z} \oplus b\mathbb{Z}$	\mathbb{Z} -module	21
$N(\mathfrak{a})$	Norm of the ideal \mathfrak{a}	21
(Q, P)	Ideal generated by Q/σ and $(P + \sqrt{D})/\sigma$	22
\mathfrak{A}_i	Matrix corresponding to the ideal (Q_i, P_i)	26
\mathfrak{a}_i	Ideal $\mathfrak{a}_i = (Q_i, P_i)$	26
α_i	Quadratic irrational corresponding to the ideal (Q_i, P_i)	27
\mathcal{A}	Set of matrices \mathfrak{A}_i for a fixed $K = \mathbb{Q}(\sqrt{D})$	26
Ψ_i	Relative generator such that $\mathfrak{a}_i = (\Psi_i)\mathfrak{a}_0$	26
R	$R = (P^2 - D)/Q$ where (Q, P) is an ideal	26
$\rho(Q_i, P_i)$	One continued fraction expansion step on the ideal	36

Chapter 1

Introduction

1.1 Ideal Arithmetic

In 1801, a man who is unquestionably one of the greatest mathematicians of all time, published a Latin work which is one of the most brilliant documents in number theory. The man was the German mathematician Karl Friedrich Gauss (1777-1855) and the work was the *Disquisitiones Arithmeticae*.

One of his many considerable contributions detailed in the book is the first rigorous explanation of the theory of binary quadratic forms. These pleasing mathematical objects are simple and yet profound. Algebraic number theory was developing at about the same time and binary quadratic forms were soon seen to be directly related to ideals of quadratic number fields. This relation combines the magnificence of abstract algebra with the elegance of number theory into a fascinating package.

A quadratic number field is an extension of the field of rational numbers which is formed by adjoining the square-root of an integer (which is not a perfect square)

to the field. The rationals contain a special subset of numbers, namely the rational integers. The concept of an integer may be extended to quadratic number fields and, as with the rational integers, these integers have the algebraic structure of a ring.

In this thesis we concentrate on ideals of the ring of integers. Although the concepts discussed here are applicable to real and imaginary quadratic fields, our implementations specifically focus on the real case. Reduction in the imaginary case has already been analysed and in fact, the reader will soon notice that the imaginary case may be treated as a simplification of the real case.

Objects in our physical world are regularly grouped into equivalence classes. For example, sorting buttons into containers by colour, or soda pop onto shelves by size. Ideals are grouped into equivalence classes as well and when we are given an ideal, our goal is to find an ideal in the same equivalence class which is represented by small parameters. In addition, we require an algebraic number which, when multiplied by the original ideal, yields an ideal with small parameters. This process is known as reduction and our goal is to find the most efficient algorithms implementing it.

Ideals of quadratic fields have many applications including finding solutions to the Pell equation, calculating the fundamental unit and regulator of a real quadratic field, factoring integers, and cryptographic key agreement protocols. These applications involve ideal multiplication. When ideals are multiplied, the parameters characterizing them often double in bit size. In this thesis we compare several algorithms that find a reduced ideal equivalent to this product.

In an application such as cryptographic key exchange, the efficiency of the reduction procedure is extremely important. Without good techniques, reduction consumes an overwhelming share of the computing time. We will see that the best approach

is one introduced by Shanks which in effect performs the reduction before the ideals are even multiplied.

To reach this conclusion, we have generalized all of the binary quadratic form reduction algorithms, most for the first time, to work with ideals of real quadratic fields. Since in this case, we have a cycle of reduced ideals rather than a unique reduced representative, it was necessary to develop a technique that computes the relative generator so that a representative from the cycle can be chosen. In practice, it is infeasible to compute the relative generator exactly; consequently, the algorithms have been implemented using the (f, p) representation of Jacobson, Scheidler and Williams which provides an approximation of the relative generator with guaranteed numerical accuracy.

This is the first comparison of ideal reduction in real quadratic fields and the most complete look at general reduction in quadratic fields. The breadth of the research provides a comprehensive understanding of the state of reduction theory and has led to the development of a very efficient, elegant, new algorithm based on the ideas of several of the techniques presented in this work. This new reduction algorithm is the fastest practical method for reducing ideals in quadratic fields when multiplication is not required.

In order to determine which algorithms work better in practice, a careful implementation of the algorithms was written in the C programming language utilizing the GNU multi-precision library. The calculations were precisely optimized to ensure that any duplication of computing effort was eliminated. This is the first implementation of many of these algorithms and the most extensive comparison to date of reduction algorithms for ideals of real quadratic fields. The library is available in the

code repository of the Centre for Information Security and Cryptography (CISaC) at the University of Calgary.

1.2 Organization Of The Thesis

We begin in Chapter 2 with an introduction to the theory of algebraic number fields. The concepts are introduced in terms of arbitrary degree number fields and then specialized to quadratic number fields where appropriate. We formally introduce the ring of integers, ideals, fractional ideals and class group. Most of the notation used throughout the thesis is introduced here.

From Chapter 3 onward we work exclusively with quadratic number fields. Chapter 3 introduces multiplication and covers reduction in detail. The relative generator is discussed and a method is developed which may be used to calculate it from within any reduction algorithm. The classical reduction algorithms of Lagrange and Gauss are presented, both theoretically, utilizing matrices and ideals, and practical implementations. When working with imaginary quadratic fields, a unique reduced ideal is obtained without any additional effort; however, real quadratic fields require choosing a representative from the cycle of reduced ideals. With the exception of this procedure, all of the concepts in this chapter apply to both real and imaginary quadratic fields.

One application of ideals is to the exchange of a secret cryptographic key by two parties across public communication channels. In Chapter 4, concepts of secure communication are explained and a key agreement protocol is outlined. Thousands of reductions are used in this process; accordingly, it provides an excellent test-bed

for comparing ideal reduction algorithms.

Chapter 5 contains the bulk of the research in this thesis. The algorithms of Rickert, Schönhage and Schnorr-Seysen are presented in terms of ideals for the first time. The Schnorr-Seysen algorithm is reworked into the language of continued fractions. With this presentation, it is easy to see its relation to some of the other algorithms. Shanks' method of reducing before multiplying, which he dubbed NUCOMP, is explained along with improvements by many individuals. A new algorithm using continued fractions in a similar style to Schnorr-Seysen and modern NUCOMP is introduced.

Practical implementations and timings of most of the algorithms are presented in Chapter 6. General programming concepts in addition to Lehmer's method of computing the GCD open the chapter. We continue with comments on implementation and finish with several tables and graphs comparing the reduction algorithms' performance during a key exchange.

Finally, concluding remarks on the algorithms are given and topics for further research are suggested.

Chapter 2

Algebraic Number Theory

Concepts

In this chapter we present some of the basic theory of algebraic number fields and describe how these general concepts specifically relate to the particular case of quadratic number fields. We introduce the ring of integers, ideals of this ring, fractional ideals and the class group. This chapter also serves to set much of the notation that will be used throughout the thesis. Most of the proofs of statements in this chapter are readily available and so are not presented here. Some example sources are, [Fra98] for abstract algebra, [ST87] for algebraic number theory and [WW87] for ideals of orders of quadratic fields.

2.1 Algebraic Number Fields

Before introducing some of the core concepts of algebraic number theory we first give a brief review of field theory. Recall that a *field* is comprised of a set of elements

along with two operations. For our purposes, it is natural to denote the operations as $+$ and \cdot which respectively represent addition and multiplication. In a field, one of the elements from the set is the additive identity (usually called zero and denoted 0) and another is the multiplicative identity (usually called one and denoted 1). The field is closed, associative, and commutative under both operations. Each element of the field has an additive inverse and the non-zero elements have a multiplicative inverse. Finally, multiplication distributes over addition.

A field E is an *extension field* of a field F if E and F share the same operations and $F \subseteq E$. E is a *finite extension* of degree n over F if E is of finite dimension n as a vector space over F . We denote the degree of E over F by $[E : F]$. If we adjoin a single element (not in F) to F , we have a *simple extension* of F . The base field typically considered in algebraic number theory is \mathbb{Q} , the field of rational numbers.

Theorem 2.1. *Every finite extension of the field \mathbb{Q} is a simple extension.*

In other words, the extension field $E = \mathbb{Q}(\alpha_1, \alpha_2, \dots, \alpha_n)$ for some finite number n is equivalent to the extension field $\mathbb{Q}(\alpha)$ for some α . One can see quite easily that α will be an element of the field E .

As stated earlier, we are interested in algebraic number fields. Each word here has significance.

Definition 2.2. An element α is *algebraic* over a field F if α is the zero of a non-zero polynomial with coefficients in F . That is, $f(\alpha) = 0$ for some non-zero polynomial $f(x) \in F[x]$.

Definition 2.3. An element of \mathbb{C} , the field of complex numbers, that is algebraic over \mathbb{Q} is called an *algebraic number*.

Example 2.4. $\sqrt{6}$ is an algebraic number since it is a zero of the polynomial $f(x) = x^2 - 6$; however, it can be shown that we cannot find a polynomial with coefficients in \mathbb{Q} such that $f(\pi) = 0$, therefore, π is not an algebraic number.

Definition 2.5. We say K is an *algebraic number field* if K is a subfield of \mathbb{C} and the degree of K over \mathbb{Q} is finite.

The integers \mathbb{Z} are the building blocks of \mathbb{Q} . By construction, algebraic numbers are fundamentally related to \mathbb{Q} and an analogue of the rational integers exists for algebraic number fields. We may generalize the concept of an integer according to the following definition.

Definition 2.6. A complex number α is an *algebraic integer* if it is a zero of a monic polynomial with coefficients in \mathbb{Z} . That is, α is an algebraic integer if $\alpha \in \mathbb{C}$ and there exists a polynomial

$$f(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0 \quad (a_i \in \mathbb{Z})$$

such that $f(\alpha) = 0$.

Example 2.7. $\sqrt{-1}$ is an algebraic integer because it is a zero of $f(x) = x^2 + 1$. The algebraic number $\frac{1}{2}$ is a zero of polynomials such as $f(x) = 2x - 1$ and $f(x) = x - \frac{1}{2}$; however, the first is not monic and the second does not have all coefficients in \mathbb{Z} . One can easily prove that a polynomial with the required properties does not exist and so $\frac{1}{2}$ is not an algebraic integer.

It is important to know the structure of the set of algebraic integers. As in the case of \mathbb{Z} , multiplicative inverses of the elements do not exist in general. Hence, the algebraic integers do not form a field; however, they do form a ring, just as \mathbb{Z} does.

Theorem 2.8. *The algebraic integers of a number field K form a ring, denoted \mathcal{O}_K (or just \mathcal{O} if the context is clear), and referred to as the ring of integers of K . Further, \mathcal{O}_K is a free \mathbb{Z} -module of rank $[K : \mathbb{Q}]$.*

Since a number field K is defined to be a finite extension of \mathbb{Q} , by Theorem 2.1 we know that $K = \mathbb{Q}(\alpha)$ for some algebraic number $\alpha \in K$. Now we will show that we may consider α to be not just an algebraic number but an algebraic integer. We first require the following lemma.

Lemma 2.9. *If K is a number field, then for any $\alpha \in K$ there exists $m \in \mathbb{Z}$ such that $m\alpha \in \mathcal{O}$.*

Proof. Let $\alpha \in K$ and $[K : \mathbb{Q}] = n$, then α is a zero of a polynomial of the form

$$f(x) = \frac{a_0}{b_0} + \frac{a_1}{b_1}x + \cdots + \frac{a_n}{b_n}x^n \quad (a_i, b_i \in \mathbb{Z}) .$$

Note that we may assume that all of the coefficients are integers since if $k = \text{lcm}[b_1, b_2, \dots, b_n]$,

$$f(\alpha) = 0 \Rightarrow kf(\alpha) = 0$$

where all coefficients of $kf(x)$ are integers. Hence, α is a zero of a polynomial

$$f(x) = a_0 + a_1x + \cdots + a_nx^n \quad (a_i \in \mathbb{Z}) .$$

Our goal now is to make the polynomial monic. Note that if we multiply the expression

by a_n^{n-1} we obtain

$$\begin{aligned} a_n^{n-1}f(x) &= a_0a_n^{n-1} + a_1a_n^{n-1}x + \cdots + a_{n-1}a_n^{n-1}x^{n-1} + a_n^n x^n \\ &= a_0a_n^{n-1} + (a_1a_n^{n-2})(a_nx) + \cdots + a_{n-1}(a_nx)^{n-1} + (a_nx)^n . \end{aligned}$$

Setting $y = a_nx$ and $a'_i = a_i a_n^{n-1-i}$ we obtain

$$a_n^{n-1}f(y/a_n) = a'_0 + a'_1y + \cdots + a'_{n-1}y^{n-1} + y^n$$

which has a zero $\alpha' = a_n\alpha$. Hence, we have

$$g(\alpha') = a'_0 + a'_1\alpha' + \cdots + a'_{n-1}(\alpha')^{n-1} + (\alpha')^n = 0 \quad (a'_i \in \mathbb{Z})$$

which is what we wanted to show. \square

Using this lemma we may prove,

Proposition 2.10. *If $K = \mathbb{Q}(\alpha)$ is a number field, then $K = \mathbb{Q}(\alpha')$ where α' is an algebraic integer.*

Proof. Let $K = \mathbb{Q}(\alpha)$ be a number field, then, as mentioned previously, α is an algebraic number. By the previous lemma, there exists $m \in \mathbb{Z}$ such that $\alpha' = m\alpha \in \mathcal{O}$. Since m is also in \mathbb{Q} , $\mathbb{Q}(\alpha) = \mathbb{Q}(m\alpha)$, and therefore $K = \mathbb{Q}(\alpha')$ with $\alpha' \in \mathcal{O}$. \square

Theorem 2.11. *If $E = F(\alpha)$ is a simple extension of a field F and $[E : F] = n$ is finite, then every element $\beta \in E$ can be uniquely written as*

$$\beta = a_0 + a_1\alpha + \cdots + a_{n-1}\alpha^{n-1} \quad (a_i \in F) .$$

By Proposition 2.10 and Theorem 2.11 every number field K has a basis $B = \{1, \alpha, \dots, \alpha^{n-1}\}$ where $\alpha \in \mathcal{O}$. Since \mathcal{O} is a ring, this basis consists entirely of algebraic integers. By Theorem 2.8 we know a \mathbb{Z} -basis exists for \mathcal{O} ; in other words, there exists a basis $\{\alpha_1, \alpha_2, \dots, \alpha_n\}$ such that for any element $\beta \in \mathcal{O}$

$$\beta = a_1\alpha_1 + a_2\alpha_2 + \dots + a_n\alpha_n \quad (a_i \in \mathbb{Z}, \alpha_i \in \mathcal{O})$$

and

$$\begin{aligned} 0 &= a_1\alpha_1 + a_2\alpha_2 + \dots + a_n\alpha_n \\ \Rightarrow a_i &= 0 \quad (1 \leq i \leq n) . \end{aligned}$$

However, it need not be the case that B is a \mathbb{Z} -basis for \mathcal{O} . For example, consider $K = \mathbb{Q}(\sqrt{5})$, then $B = \{1, \sqrt{5}\}$ is a basis for K but $\frac{1+\sqrt{5}}{2}$ is an integer in K since it is a zero of the polynomial $x^2 - x - 1 = 0$. Yet $\frac{1+\sqrt{5}}{2} \neq a + b\sqrt{5}$ for any $a, b \in \mathbb{Z}$, hence B is not a \mathbb{Z} -basis for \mathcal{O} .

Definition 2.12. Let K be a number field of degree n over \mathbb{Q} and let $B = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$ be a \mathbb{Z} -basis for \mathcal{O} . Then B is an *integral basis* for K .

An important invariant of K is the discriminant.

Theorem 2.13. *If α is algebraic over a field K , then there exists a unique monic polynomial f of minimal degree with rational coefficients such that $f(\alpha) = 0$. We say f is the minimal polynomial of α over K .*

Theorem 2.14. *Let $K = \mathbb{Q}(\alpha)$ be a number field of degree n over \mathbb{Q} . Then there*

exist n distinct monomorphisms $\lambda_i : K \rightarrow \mathbb{C}$ ($1 \leq i \leq n$). Let $\alpha_i = \lambda_i(\alpha)$, then the α_i are the distinct zeros of the minimal polynomial of α over \mathbb{Q} .

Definition 2.15. For any element α of a number field K , the $\lambda_i(\alpha)$ are the K -conjugates of α .

Definition 2.16. Let $\{\alpha_1, \alpha_2, \dots, \alpha_n\}$ be an integral basis for a number field K , then the *discriminant* of K , denoted Δ_K (or just Δ if the context is clear), is

$$\Delta = \begin{vmatrix} \lambda_1(\alpha_1) & \lambda_1(\alpha_2) & \cdots & \lambda_1(\alpha_n) \\ \lambda_2(\alpha_1) & \lambda_2(\alpha_2) & \cdots & \lambda_2(\alpha_n) \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_n(\alpha_1) & \lambda_n(\alpha_2) & \cdots & \lambda_n(\alpha_n) \end{vmatrix}^2.$$

Quadratic Number Fields

With the exception of \mathbb{Q} , quadratic number fields are the simplest algebraic number fields. We will now discuss these concepts in this context. First, we characterize all quadratic number fields.

Proposition 2.17. *Number fields of degree 2 over \mathbb{Q} are precisely of the form $\mathbb{Q}(\sqrt{D})$ for square-free integers D .*

Proof. Let $K = \mathbb{Q}(\alpha)$ be a degree 2 number field over \mathbb{Q} . By Proposition 2.10, we may assume $\alpha \in \mathcal{O}$. This implies that α is a zero of a monic degree 2 polynomial $f(x) = x^2 + bx + c$ with $b, c \in \mathbb{Z}$.

By the quadratic formula,

$$\alpha = \frac{-b \pm \sqrt{b^2 - 4c}}{2}.$$

Since $b, c \in \mathbb{Z}$ we can write $b^2 - 4c = r^2 D$ with D square-free and $r, D \in \mathbb{Z}$. Thus we have

$$\alpha = \frac{-b \pm \sqrt{r^2 D}}{2} = \frac{-b \pm r\sqrt{D}}{2} = \frac{-b}{2} \pm \frac{r}{2}\sqrt{D}.$$

Since $b, r \in \mathbb{Z}$, $\frac{b}{2}, \frac{r}{2} \in \mathbb{Q}$, we have $\mathbb{Q}(\alpha) = \mathbb{Q}(\sqrt{D})$ where D is square-free. \square

Definition 2.18. Let $D \in \mathbb{Z}$ be square-free and let $K = \mathbb{Q}(\sqrt{D})$. If $D > 0$, we say K is a *real quadratic field*; if $D < 0$, we say K is an *imaginary quadratic field*.

Theorem 2.19. *The ring of integers of a quadratic field has the following characterization:*

$$\mathcal{O} = \begin{cases} \mathbb{Z}\left[\frac{1+\sqrt{D}}{2}\right] & \text{if } D \equiv 1 \pmod{4} \\ \mathbb{Z}[\sqrt{D}] & \text{if } D \equiv 2, 3 \pmod{4} \end{cases}$$

The properties of quadratic fields are often characterized by the same set of cases used above. We will simplify the notation by making use of the following definition.

Definition 2.20.

$$\sigma = \begin{cases} 2 & \text{if } D \equiv 1 \pmod{4} \\ 1 & \text{if } D \equiv 2, 3 \pmod{4} \end{cases}$$

We also set $\omega = \frac{\sigma - 1 + \sqrt{D}}{\sigma}$ and so \mathcal{O} becomes $\mathbb{Z}[\omega]$. The monomorphisms of a

quadratic field are

$$\begin{aligned}\lambda_1(a + b\sqrt{D}) &= a + b\sqrt{D} , \\ \lambda_2(a + b\sqrt{D}) &= a - b\sqrt{D} .\end{aligned}$$

We denote the conjugate of an element α by $\bar{\alpha}$. The discriminant of $\mathbb{Q}(\sqrt{D})$ is calculated to be

$$\Delta = \begin{vmatrix} 1 & \omega \\ 1 & \bar{\omega} \end{vmatrix}^2 = \left(\frac{\sigma - 1 - \sqrt{D}}{\sigma} - \frac{\sigma - 1 + \sqrt{D}}{\sigma} \right)^2 = \left(\frac{2}{\sigma} \right)^2 D .$$

2.2 Ideals

An algebraic number field does not necessarily have unique factorization. While trying to solve Fermat's last theorem, Kummer devised the notion of *ideal numbers*. Using ideal numbers, he could again obtain a unique factorization and this helped him to solve Fermat's last theorem for a wide class of prime exponents. Dedekind extended the power of ideal numbers by generalizing the concept to arbitrary rings.

Definition 2.21. Let R be a ring, then A is an *ideal* of R if for every $a, b \in A$ and $r \in R$ we have $a - b \in A$ and $ar \in A$. Informally, an ideal is a subring with the additional property of absorption.

Note. Recall from Theorem 2.8 that \mathcal{O} forms a ring, thus we may consider the ideals of \mathcal{O} . By convention, we represent the ideals of \mathcal{O} using Gothic notation. Following the notation of Hungerford [Hun80], we denote the fact that the ideal \mathfrak{a} is a subring of \mathcal{O} by $\mathfrak{a} \subseteq \mathcal{O}$.

As in the case of \mathcal{O} , we wish to impose some algebraic structure on the set of ideals. We would like to form a group and so we need to define a binary operation among ideals.

Definition 2.22. Let \mathfrak{a} and \mathfrak{b} be ideals; we define *ideal multiplication* to be

$$\mathfrak{a}\mathfrak{b} = \left\{ \sum a_i b_i \mid a_i \in \mathfrak{a}, b_i \in \mathfrak{b}, i \in \mathbb{N} \right\}.$$

In words, the product of two ideals is the set of all finite sums of the products of elements of \mathfrak{a} and \mathfrak{b} .

It can be shown that ideals are closed under multiplication. That is, the product of two ideals is again an ideal. It is also not hard to apply the definitions and see that they are associative, commutative, and have identity element \mathcal{O} , and hence form a monoid. Unfortunately, we fail when we try to find element inverses. Using \mathbb{Z} as an example, it is immediately clear that given an ideal $m\mathbb{Z}$, we are not be able to find an ideal $n\mathbb{Z}$ (with $m, n \in \mathbb{Z}, m \neq 0, 1$) so that $(m\mathbb{Z})(n\mathbb{Z}) = mn\mathbb{Z} = \mathbb{Z}$. Hence, we need to introduce an analogue to the rational numbers.

Definition 2.23. We define \mathfrak{a} to be a *fractional ideal* of \mathcal{O} if

$$\mathfrak{a} = c^{-1}\mathfrak{b} \quad \text{for some (ordinary) ideal } \mathfrak{b} \text{ and non-zero } c \in \mathcal{O}.$$

To avoid confusion, an ideal that is not a fractional ideal is sometimes called an *integral ideal*.

Example 2.24. To find the fractional ideals of \mathbb{Z} we take $\mathcal{O} = \mathbb{Z}$ in the above

definition and obtain

$$\begin{aligned} \mathfrak{a} &= c^{-1}\mathfrak{b} = c^{-1}(b\mathbb{Z}) & (b, c \in \mathbb{Z}, c \neq 0) \\ &= \frac{b}{c}\mathbb{Z} = q\mathbb{Z} & (q \in \mathbb{Q}) . \end{aligned}$$

Again, we may apply the definitions and see that fractional ideals are associative, commutative, and have identity element \mathcal{O} . The definition of fractional ideals was designed to make inverses exist, the construction of which is given by the following proposition.

Theorem 2.25. *Let K be a number field with ring of integers \mathcal{O} and let \mathfrak{a} be a non-zero fractional ideal, then \mathfrak{a} has inverse*

$$\mathfrak{a}^{-1} = \{c \in K \mid c\mathfrak{a} \subseteq \mathcal{O}\}$$

yielding $\mathfrak{a}\mathfrak{a}^{-1} = \mathfrak{a}^{-1}\mathfrak{a} = \mathcal{O}$.

With this result in hand we are able to state the structure of the fractional ideals of \mathcal{O} .

Theorem 2.26. *The non-zero fractional ideals of a ring of integers \mathcal{O} form an abelian group with identity element \mathcal{O} under the operation of ideal multiplication. The group of fractional ideals of \mathcal{O} is denoted by \mathcal{F} .*

As we saw by the definitions of ideal inverses and multiplication, ideals can have quite a complex description. On the other hand, there is a special type of ideal that requires only a single element to describe it. We first introduce the concept of generators.

Definition 2.27. Let X be a subset of \mathcal{O} . We denote by $\mathfrak{a} = (X)$ the smallest ideal of \mathcal{O} that contains X . The elements of X are called the *generators* of \mathfrak{a} . If X is finite, consisting of the elements $\alpha_1, \alpha_2, \dots, \alpha_n$, we may write $\mathfrak{a} = (\alpha_1, \alpha_2, \dots, \alpha_n)$.

Definition 2.28. An ideal of a ring R with a multiplicative identity is called *principal* if it is generated by a single element of R . In symbols, the ideal generated by a is denoted by (a) and

$$(a) = \{ar \mid r \in R\} .$$

Example 2.29. If we take $\mathcal{O} = \mathbb{Z}[i]$,

$$\begin{aligned} (3) &= \{3r \mid r \in \mathbb{Z}[i]\} \\ &= \{3(a + bi) \mid a, b \in \mathbb{Z}\} \end{aligned}$$

is a principal ideal.

Definition 2.30. The ideal (0) is the *zero ideal*.

The following remarkable result shows that any ideal of \mathcal{O} can be generated with just two elements.

Theorem 2.31. Let \mathfrak{a} be a non-zero ideal of \mathcal{O} and $\alpha \neq 0$ an element of \mathfrak{a} , then there exists $\beta \in \mathfrak{a}$ such that $\mathfrak{a} = (\alpha, \beta)$.

Definition 2.32. A fractional ideal \mathfrak{a} of \mathcal{O} is a *principal fractional ideal* if $\mathfrak{a} = c^{-1}\mathfrak{b}$ for some principal (integral) ideal $\mathfrak{b} = (b)$ and non-zero $c \in \mathcal{O}$.

Using our previous notation, $\mathfrak{a} = (b/c)$ with $b, c \in \mathcal{O}$, $c \neq 0$. One readily observes that the set of principal fractional ideals \mathcal{P} is a subset of \mathcal{F} and also that \mathcal{P} is closed under multiplication. These facts classify \mathcal{P} as a subgroup of \mathcal{F} .

Notice that the only difference in the definition of fractional ideals and principal fractional ideals is that \mathfrak{b} is required to be principal. The difference between principal ideals and principal fractional ideals is the denominator c in the generator. This idea extends to the generators of non-principal fractional ideals as well.

2.3 Class Group and Class Number

We know from group theory that every subgroup of an abelian group is also abelian and so \mathcal{P} is abelian. Some subgroups have a special property that is exhibited in a relationship between the elements of the parent group and the subgroup itself.

Definition 2.33. A subgroup H of a group G is called *normal* if $gH = Hg$ for all $g \in G$.

Theorem 2.34. *If G is abelian, every subgroup H of G is normal.*

Theorem 2.35. *If G is a group and H is a normal subgroup of G , then the set of cosets*

$$\{gH \mid g \in G\}$$

is a group under the operation $(aH)(bH) = abH$ if and only if H is normal. This group is called a factor group and is denoted G/H .

All of the components required for a factor group are available to us. Using the abelian group \mathcal{F} , and \mathcal{P} a normal subgroup of \mathcal{F} , we define the class group.

Definition 2.36. Let \mathcal{F} be the abelian group of fractional ideals and \mathcal{P} the normal subgroup of principal fractional ideals. The *class group* of \mathcal{O} , denoted \mathcal{H} , is defined to be the factor group

$$\mathcal{H} = \mathcal{F}/\mathcal{P} .$$

We will show that \mathcal{H} is always finite. The order of \mathcal{H} is referred to as the *class number* and denoted h .

By the definition of a factor group,

$$\mathcal{H} = \{a\mathcal{P} \mid a \in \mathcal{F}\}$$

and so we see that \mathcal{H} partitions the elements of \mathcal{F} into cosets. If $a \in \mathcal{P}$ then $a\mathcal{P}$ is simply \mathcal{P} (\mathcal{P} is closed under multiplication) so all principal ideals of \mathcal{F} are gathered into one of the cosets of \mathcal{H} . If every ideal of \mathcal{F} is principal, we have $\mathcal{F} = \mathcal{P}$, thus \mathcal{P} is the only element of \mathcal{H} and $h = 1$.

Definition 2.37. Two fractional ideals \mathfrak{a} and \mathfrak{b} are said to be *equivalent* if they map to the same element of \mathcal{H} . This relationship is denoted $\mathfrak{a} \sim \mathfrak{b}$ and the equivalence class of \mathfrak{a} is denoted $[\mathfrak{a}]$.

This tells us that if $\mathfrak{a} \sim \mathfrak{b}$ then $\left(\frac{\psi_1}{\psi_2}\right)\mathfrak{a} = \mathfrak{b}$ where $\psi_1, \psi_2 \in \mathcal{O}$, $\psi_2 \neq 0$, and $\left(\frac{\psi_1}{\psi_2}\right)$ is a principal fractional ideal. Alternatively, $(\psi_1)\mathfrak{a} = (\psi_2)\mathfrak{b}$ where (ψ_1) and (ψ_2) are principal ideals.

Definition 2.38. Let \mathfrak{a} and \mathfrak{b} be equivalent ideals. The term *relative generator* refers to the number ψ generating a principal fractional ideal such that $(\psi)\mathfrak{a} = \mathfrak{b}$.

The definition of equivalence was given in terms of fractional ideals but the following theorem gives an important relationship to ideals.

Theorem 2.39. *Every equivalence class contains an integral ideal. Consequentially, every fractional ideal is equivalent to an integral ideal.*

Proof. Let $\mathfrak{a} \in \mathcal{F}$, then

$$\begin{aligned} \mathfrak{a} &= c^{-1}\mathfrak{b} & (c \neq 0 \in \mathcal{O}, \mathfrak{b} \subseteq \mathcal{O}) \\ \Rightarrow c\mathfrak{a} &= \mathfrak{b} \\ \Rightarrow (c)\mathfrak{a} &= \mathfrak{b} . \end{aligned}$$

Now since $(c) \in \mathcal{P}$, \mathfrak{a} is equivalent to \mathfrak{b} . Given that \mathfrak{a} was arbitrary, every equivalence class contains an ideal. \square

Rings have a parallel to factor groups, the concept of a factor ring. To form a factor group, the subgroup was required to be normal. Similarly, factor rings require that the subring be an ideal.

Theorem 2.40. *If R is a ring and A is a subring of R , then the set of cosets*

$$\{r + A \mid r \in R\}$$

is a ring under the operations $(a + A) + (b + A) = a + b + A$ and $(a + A)(b + A) = ab + A$ if and only if A is an ideal. This ring is called a factor ring and is denoted R/A .

It can be shown that if \mathfrak{a} is a non-zero ideal then the factor ring \mathcal{O}/\mathfrak{a} is finite, hence we can make the following definition.

Definition 2.41. The *norm* of a non-zero ideal \mathfrak{a} is defined to be the order of the factor ring of \mathcal{O} by \mathfrak{a} . In symbols,

$$N(\mathfrak{a}) = |\mathcal{O}/\mathfrak{a}|.$$

The concept of \mathbb{Z} -bases apply to ideals also and we have the following important fact.

Theorem 2.42. *Let K be a number field of degree n over \mathbb{Q} , then any non-zero ideal \mathfrak{a} of \mathcal{O} has an n element \mathbb{Z} -basis.*

Let $\omega = \frac{\sigma - 1 + \sqrt{D}}{\sigma}$ as before. Each non-zero ideal \mathfrak{a} of the ring of integers of a quadratic field may be written uniquely as the \mathbb{Z} module

$$\mathfrak{a} = a\mathbb{Z} \oplus (b + c\omega)\mathbb{Z}$$

where $a, b, c \in \mathbb{Z}$, $a, c > 0$, $0 \leq b < a$, $c|a$, $c|b$, $ac|N(b + c\omega)$. Setting $Q = a\sigma/c$, $P = b\sigma/c + \sigma - 1$, $S = c$ we see that this is the same as

$$\mathfrak{a} = S\frac{Q}{\sigma}\mathbb{Z} \oplus S\left(\frac{P + \sqrt{D}}{\sigma}\right)\mathbb{Z}$$

where Q/σ , P , $S \in \mathbb{Z}$, $\sigma Q|P^2 - D$ and $Q, P > 0$. Q/σ is the least positive rational integer in \mathfrak{a} ; the norm of \mathfrak{a} is easily calculated as SQ/σ . Since the only variables in this representation are Q , P and S , we use the notation $(S)(Q, P)$ to represent the ideal.

Definition 2.43. An ideal is *primitive* if it has no rational prime divisors.

We will usually work with ideals where $S = 1$. In this case, by the structure of \mathcal{O} and the fact that Q/σ is the least element in \mathfrak{a} , the ideal is primitive, and we simply write (Q, P) .

To prove the class group is finite, we require that for any given norm, there are only a finite number of ideals of that norm. This comes as a corollary to the following two theorems. We will state the first and prove the second.

Theorem 2.44. *Any integer belongs to only a finite number of ideals of \mathcal{O} .*

Theorem 2.45. *For any non-zero ideal \mathfrak{a} , $N(\mathfrak{a}) \in \mathfrak{a}$.*

Proof. Let \mathfrak{a} be a non-zero ideal of \mathcal{O} and let $x \in \mathcal{O}$. Since $N(\mathfrak{a})$ is the order of \mathcal{O}/\mathfrak{a} ,

$$\begin{aligned} N(\mathfrak{a})(x + \mathfrak{a}) &= 0 + \mathfrak{a} && (0 + \mathfrak{a} \text{ is the additive identity}) \\ \Rightarrow N(\mathfrak{a})(1 + \mathfrak{a}) &= 0 + \mathfrak{a} && (\text{setting } x = 1 \in \mathcal{O}) \\ \Rightarrow N(\mathfrak{a}) + \mathfrak{a} &= 0 + \mathfrak{a} \\ \Rightarrow N(\mathfrak{a}) &\in \mathfrak{a} . && \square \end{aligned}$$

Corollary 2.46. *Only finitely many ideals have a given norm.*

Proof. By Theorem 2.44 there are only a finite number of ideals that contain the integer $N(\mathfrak{a})$ and by Theorem 2.45, every ideal has its norm as an element. Therefore, only a finite number of ideals can have a given norm. \square

With one more fact, we can prove that the order of the class group is finite. The proof of the following lemma uses a theorem of Minkowski relating to lattices and a

tighter bound than we claim here is possible. Our bound reflects only the concepts we have introduced and any fixed bound would do. We state the result here as a lemma and refer the interested reader to [ST87].

Lemma 2.47. *Every non-zero ideal is equivalent to an ideal with norm at most $\sqrt{|\Delta|}$.*

Theorem 2.48. *The class group \mathcal{H} forms a finite abelian group.*

Proof. The fact that \mathcal{H} is abelian follows from the fact that \mathcal{F} and \mathcal{P} are abelian.

To show that \mathcal{H} is finite, let $[\mathfrak{a}] \in \mathcal{H}$ and let Δ be the discriminant of K . By Theorem 2.39, $\mathfrak{a} \sim \mathfrak{b}$ where \mathfrak{b} is an integral ideal. By Lemma 2.47, there exists an integral ideal \mathfrak{c} with norm at most $\sqrt{|\Delta|}$ where $\mathfrak{b} \sim \mathfrak{c}$. Hence, $[\mathfrak{a}] = [\mathfrak{b}] = [\mathfrak{c}]$. Now by Corollary 2.46, only a finite number of ideals have a given norm. This means that for each i from 1 to $\left\lfloor \sqrt{|\Delta|} \right\rfloor$, there are a finite number of ideals with norm i . Summing together, we obtain only a finite number of ideals that have the properties of \mathfrak{c} . Since every equivalence class contains an ideal like \mathfrak{c} , the number of equivalence classes is finite. That is, \mathcal{H} is finite. \square

This theorem tells us a tremendous amount of information about the structure of \mathcal{H} . For example, the Fundamental Theorem of Finite Abelian Groups says that any finite abelian group is isomorphic to the direct product of cyclic groups of prime power order. Therefore,

$$\mathcal{H} \approx \mathbb{Z}_{p_1^{a_1}} \oplus \mathbb{Z}_{p_2^{a_2}} \oplus \cdots \oplus \mathbb{Z}_{p_n^{a_n}}$$

where $a_i, p_i \in \mathbb{N}$ and the p_i are (not necessarily distinct) primes. Additionally, this factorization is unique up to the ordering of the factors.

Chapter 3

Multiplication and Reduction

Theory

From this point on, we restrict our focus to ideals of quadratic fields with particular emphasis on real quadratic fields. Throughout, we assume the field $K = \mathbb{Q}(\sqrt{D})$ is fixed where D is square-free.

In this chapter, we begin with an overview of ideal multiplication and present the algorithms of Arndt and Shanks. We determine the equivalence class of an ideal and provide a method of calculating the principal ideal relating two equivalent ideals. Reduction is described and the classical algorithms of Lagrange and Gauss are presented. An approach to choosing a unique reduced representative that is equivalent to the product of two ideals is explained. As well, a method is introduced which converts algorithms for working with forms to those for ideals. This chapter forms the foundation upon which the work of Chapters 5 and 6 is based.

3.1 Multiplication

In Definition 2.22, ideal multiplication was defined as

$$\mathfrak{a}\mathfrak{b} = \left\{ \sum a_i b_i \mid a_i \in \mathfrak{a}, b_i \in \mathfrak{b}, i \in \mathbb{N} \right\}.$$

While this is a useful theoretical definition, it is impossible to implement on a computer. Since we are restricting our focus to quadratic fields, the task of computing the product of two ideals is quite simple. An algorithm of Arndt is described in [Bue89] that gives the following method of computing the product of two primitive ideals of the same discriminant. To multiply $\mathfrak{a} = (Q_a, P_a)$ and $\mathfrak{b} = (Q_b, P_b)$ resulting in the product $(S)(Q_0, P_0)$, compute

$$\begin{aligned} \sigma S &= \gcd(Q_a, Q_b, P_a + P_b) = Q_a X + Q_b Y + (P_a + P_b)Z, \\ Q_0 &= \frac{Q_a Q_b}{\sigma S^2}, \\ P_0 &= \frac{1}{\sigma S} (Q_a P_b X + Q_b P_a Y + (P_a P_b + D)Z). \end{aligned}$$

The reason we need to compute the above greatest common divisor (GCD) is that even though we are multiplying two primitive ideals, their product need not be primitive. The above calculations ensure that we have $\mathfrak{a}\mathfrak{b} = (S)(Q_0, P_0)$ with (Q_0, P_0) primitive. Shanks presented the following computationally more efficient algorithm in [Sha71].

$$\begin{aligned} \sigma G &= \gcd(Q_a, Q_b) = Q_a X + Q_b Y, \\ \sigma S &= \gcd(G, P_a + P_b) = GZ + (P_a + P_b)W, \end{aligned} \tag{3.1}$$

$$\begin{aligned}
U &\equiv (P_b - P_a)XZ - R_aW \pmod{Q_b/S} , \\
Q_0 &= \frac{Q_a Q_b}{\sigma S^2} , \\
P_0 &= \frac{Q_a U}{\sigma S} + P_a .
\end{aligned}$$

Note. Throughout this thesis, if the symbol R is used in the context of an ideal (Q, P) , it always refers to the integer $(P^2 - D)/Q$. In cases where it is subscripted, the same subscripts apply to Q and P .

3.2 Equivalence

In Definition 2.37 two ideals are defined to be equivalent if they are in the same equivalence class and this in turn implies that a principal fractional ideal relates them. Here, we provide an explicit calculation of the equivalence class of an ideal and show how to compute a generator of the principal fractional ideal relating two equivalent ideals. This provides the groundwork to easily calculate the relative generator in the subsequent reduction algorithms.

3.2.1 Equivalence Classes

By Definition 2.37 we know that if two ideals $\mathfrak{a}_0 = (Q_0, P_0)$ and $\mathfrak{a}_i = (Q_i, P_i)$ are equivalent, then there exists a relative generator Ψ_i such that $\mathfrak{a}_i = (\Psi_i)\mathfrak{a}_0$. Conceptually, the computations are easier to follow if they are worked in terms of matrices. Let the matrix $\mathfrak{A}_i = \begin{pmatrix} Q_i & P_i \\ P_i & R_i \end{pmatrix}$ correspond to \mathfrak{a}_i . For a fixed $K = \mathbb{Q}(\sqrt{D})$, let the set of all such matrices be denoted by \mathcal{A} . Recall that (Q_i, P_i) is notation for the ideal generated by Q_i/σ and $(P_i + \sqrt{D})/\sigma$. It will be convenient for us to represent the

quotient of these two generators by the quadratic irrational

$$\alpha_i = \frac{(P_i + \sqrt{D})/\sigma}{Q_i/\sigma} = \frac{P_i + \sqrt{D}}{Q_i}.$$

Definition 3.1. By $\text{GL}(2, \mathbb{Z})$ we denote the *general linear* group of 2×2 integral matrices which are invertible. Over \mathbb{Z} this simply means that they have determinant ± 1 . $\text{PGL}(2, \mathbb{Z})$ is the factor group $\text{GL}(2, \mathbb{Z}) / \pm I_2$. That is, $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and $\begin{pmatrix} -a & -b \\ -c & -d \end{pmatrix}$ are equivalent in $\text{PGL}(2, \mathbb{Z})$.

By [KW90, Prop. 3i], the ideals \mathfrak{a}_0 and \mathfrak{a}_i are equivalent if and only if there exists a matrix $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \text{PGL}(2, \mathbb{Z})$ such that $\alpha_i = \frac{a\alpha_0 + b}{c\alpha_0 + d}$. Using these concepts we may compute a general formula characterizing all ideals equivalent to a given ideal.

Proposition 3.2. *The equivalence class of an ideal (Q_0, P_0) is given by*

$$[(Q_0, P_0)] = \left\{ (Q_i, P_i) \left| \begin{array}{l} Q_i = \varepsilon(d^2 Q_0 + 2cdP_0 + c^2 R_0), \\ P_i = \varepsilon(bdQ_0 + (ad + bc)P_0 + acR_0), \\ \varepsilon = ad - bc = \pm 1, \quad a, b, c, d \in \mathbb{Z} \end{array} \right. \right\}. \quad (3.2)$$

Proof. Let $\alpha_0 = \frac{P_0 + \sqrt{D}}{Q_0}$ be given, then $\alpha_i = \frac{P_i + \sqrt{D}}{Q_i}$ is equivalent to α_0 if, and only if, for some $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \text{PGL}(2, \mathbb{Z})$,

$$\begin{aligned} \alpha_i = \frac{a\alpha_0 + b}{c\alpha_0 + d} &= \frac{a\left(\frac{P_0 + \sqrt{D}}{Q_0}\right) + b}{c\left(\frac{P_0 + \sqrt{D}}{Q_0}\right) + d} = \frac{aP_0 + bQ_0 + a\sqrt{D}}{cP_0 + dQ_0 + c\sqrt{D}} \\ &= \left(\frac{aP_0 + bQ_0 + a\sqrt{D}}{cP_0 + dQ_0 + c\sqrt{D}}\right) \left(\frac{cP_0 + dQ_0 - c\sqrt{D}}{cP_0 + dQ_0 - c\sqrt{D}}\right) \end{aligned}$$

$$\begin{aligned}
&= \frac{bdQ_0^2 + (ad + bc)PQ_0 + ac(P_0^2 - D) + (ad - bc)Q_0\sqrt{D}}{d^2Q_0^2 + 2cdPQ_0 + c^2(P_0^2 - D)} \\
&= \frac{\varepsilon(bdQ_0 + (ad + bc)P_0 + acR_0) + \sqrt{D}}{\varepsilon(d^2Q_0 + 2cdP_0 + c^2R_0)} \quad (\varepsilon = ad - bc) \\
&= \frac{P_i + \sqrt{D}}{Q_i} . \quad \square
\end{aligned}$$

Compare this with the action of a matrix on \mathfrak{A}_0 . Let $N = \begin{pmatrix} e & f \\ g & h \end{pmatrix}$ with $e, f, g, h \in \mathbb{Z}$; then

$$N^T \mathfrak{A}_0 N = \begin{pmatrix} e^2Q_0 + 2egP_0 + g^2R_0 & efQ_0 + (eh + fg)P_0 + ghR_0 \\ efQ_0 + (eh + fg)P_0 + ghR_0 & f^2Q_0 + 2fhP_0 + h^2R_0 \end{pmatrix} . \quad (3.3)$$

Since Proposition 3.2 characterizes the equivalence class of an ideal, comparing the elements of M and N shows that all ideals (Q_i, P_i) equivalent to (Q_0, P_0) are given by

$$\begin{pmatrix} Q_i & P_i \\ P_i & R_i \end{pmatrix} = \varepsilon \begin{pmatrix} d & b \\ c & a \end{pmatrix}^T \begin{pmatrix} Q_0 & P_0 \\ P_0 & R_0 \end{pmatrix} \begin{pmatrix} d & b \\ c & a \end{pmatrix} .$$

We define two functions that represent these actions.

Definition 3.3. The function λ_M where $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \text{PGL}(2, \mathbb{Z})$ represents the action of the matrix M on the irrational number α_0 corresponding to \mathfrak{a}_0 .

$$\begin{aligned}
\lambda_M : \quad \mathbb{Q}(\sqrt{D}) &\rightarrow \mathbb{Q}(\sqrt{D}) \\
\frac{P_0 + \sqrt{D}}{Q_0} &\mapsto \frac{\varepsilon(bdQ_0 + (ad + bc)P_0 + acR_0) + \sqrt{D}}{\varepsilon(d^2Q_0 + 2cdP_0 + c^2R_0)} = \frac{P_i + \sqrt{D}}{Q_i}
\end{aligned}$$

with $\varepsilon = \det M$.

Definition 3.4. The function ϕ_N where $N = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \text{PGL}(2, \mathbb{Z})$ represents the action of the matrix N on the matrix \mathcal{A}_0 corresponding to \mathfrak{a}_0 .

$$\phi_N : \quad \mathcal{A} \rightarrow \mathcal{A}$$

$$\begin{pmatrix} Q_0 & P_0 \\ P_0 & R_0 \end{pmatrix} \mapsto \varepsilon N^T \begin{pmatrix} Q_0 & P_0 \\ P_0 & R_0 \end{pmatrix} N = \begin{pmatrix} Q_i & P_i \\ P_i & R_i \end{pmatrix}$$

with $\varepsilon = \det N$.

A natural question is to inquire about the structure of these functions. The proof that $\Lambda = \{\lambda_M \mid M \in \text{PGL}(2, \mathbb{Z})\}$ and $\Phi = \{\phi_N \mid N \in \text{PGL}(2, \mathbb{Z})\}$ form groups under function composition with identity elements ϕ_I and λ_I , where $I = \pm I_2$, is straightforward. Composition in Λ is given by $\lambda_{M_1} \circ \lambda_{M_2} = \lambda_{M_1 M_2}$ while composition in Φ is given by $\phi_{N_1} \circ \phi_{N_2} = \phi_{N_2 N_1}$.

In the course of this thesis, we move seamlessly between transformations of irrationals corresponding to ideals, and transformations of matrices corresponding to ideals, so it is important to establish that these groups of functions are in fact isomorphic. This way we know that for any transformation acting on the matrix corresponding to an ideal, there is a unique transformation acting on the irrational corresponding to the same ideal. For example, if the groups were not isomorphic, we could not easily calculate the relative generator of an ideal obtained using $\mathcal{A}_i = \phi_N(\mathcal{A}_0)$.

Proposition 3.5. *Let Λ and Φ be as defined in Definitions 3.3 and 3.4. Then*

$$\begin{aligned} \Theta : \quad & \Lambda \rightarrow \Phi \\ & \lambda_M \mapsto \phi_N \\ \text{defined by} \quad & M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \mapsto N = \begin{pmatrix} d & b \\ c & a \end{pmatrix} \end{aligned}$$

is a group isomorphism.

Proof. Let $M, M_1, M_2, N, N_1, N_2, N' \in \text{PGL}(2, \mathbb{Z})$, $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, and $N = \begin{pmatrix} d & b \\ c & a \end{pmatrix}$.

Well-defined: Observe that $\lambda_M = \lambda_{-M}$ and $\phi_N = \phi_{-N}$. Since Λ and Φ form groups, the functions λ and ϕ comprising them are bijections. Hence, $\lambda_{M_1} = \lambda_{M_2} \Rightarrow M_2 = \pm M_1$. Also,

$$\Theta(\lambda_M) = \phi_N = \phi_{-N} = \Theta(\lambda_{-M}) .$$

Hence, $\lambda_{M_1} = \lambda_{M_2} \Rightarrow \Theta(\lambda_{M_1}) = \Theta(\lambda_{M_2})$ and so Θ is well-defined.

Operation preserving: Let $M_i = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix}$ and $N_i = \begin{pmatrix} d_i & b_i \\ c_i & a_i \end{pmatrix}$ then

$$\Theta(\lambda_{M_1} \lambda_{M_2}) = \Theta(\lambda_{M_1 M_2}) = \phi_{N'}$$

where $N' = \begin{pmatrix} c_1 b_2 + d_1 d_2 & a_1 b_2 + b_1 d_2 \\ c_1 a_2 + d_1 c_2 & a_1 a_2 + b_1 c_2 \end{pmatrix}$. Similarly,

$$\Theta(\lambda_{M_1}) \Theta(\lambda_{M_2}) = \phi_{N_1} \phi_{N_2} = \phi_{N_2 N_1} = \phi_{N'}$$

Injective: We will calculate $\text{Ker}(\Theta) = \{\lambda_M \mid \Theta(\lambda_M) = \phi_I\}$. Let M and N be as given

above and set $\varepsilon = \det M$.

$$\begin{aligned}
& \Theta(\lambda_M) = \phi_I \\
& \Rightarrow \phi_N = \phi_I \\
& \Rightarrow \varepsilon \begin{pmatrix} d^2Q + 2cdP + c^2R & bdQ + (ad + bc)P + acR \\ bdQ + (ad + bc)P + acR & b^2Q + 2abP + a^2R \end{pmatrix} = \begin{pmatrix} Q & P \\ P & R \end{pmatrix} \\
& \Rightarrow \varepsilon = 1, \ a = d = \pm 1, \text{ and } b = c = 0 \\
& \Rightarrow M = N = I \\
& \Rightarrow \text{Ker}(\Theta) = \{\lambda_I\} \\
& \Rightarrow \Theta \text{ is injective}
\end{aligned}$$

Surjection: Let ϕ_N be given and let M, N be as above, then $\Theta(\lambda_M) = \phi_N$. \square

3.2.2 Relative Generator

Let us write $\alpha_i = \lambda_M(\alpha_0)$ semantically as

$$\begin{pmatrix} \alpha_i \\ 1 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \alpha_0 \\ 1 \end{pmatrix}.$$

By [KW90, Prop. 3ii], the relative generator is $\Psi_i = \varepsilon(d + c\bar{\alpha}_0)$ and so $(Q_i, P_i) = (\Psi_i)(Q_0, P_0)$. Writing λ_M in this form shows clearly that

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} \begin{pmatrix} \alpha_i \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_0 \\ 1 \end{pmatrix}.$$

In other words, $(\lambda_M)^{-1} = \lambda_{M^{-1}}$. Calculating Ψ_i^{-1} we find

$$\begin{aligned} \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} &= \varepsilon \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \\ \Rightarrow \Psi_i^{-1} &= \varepsilon^2(a - c\bar{\alpha}_i) = a - c\bar{\alpha}_i . \end{aligned}$$

Putting everything together, any algorithm in this thesis can be thought to use $\phi_N \in \Phi$ to produce a reduced ideal (Q_i, P_i) . If $N = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ then $\alpha_i = \lambda_M(\alpha_0)$ where $M = \begin{pmatrix} d & b \\ c & a \end{pmatrix}$. In this case,

$$\begin{aligned} \Psi_i &= \varepsilon(a + c\bar{\alpha}_0) = \varepsilon \left(a + c \frac{P_0 - \sqrt{D}}{Q_0} \right) , \\ \Psi_i^{-1} &= d - c\bar{\alpha}_i = d - c \frac{P_i - \sqrt{D}}{Q_i} . \end{aligned}$$

To be explicit,

$$(Q_i, P_i) = \left(\varepsilon a + \varepsilon c \frac{P_0 - \sqrt{D}}{Q_0} \right) (Q_0, P_0) , \quad (3.4)$$

$$(Q_0, P_0) = \left(d - c \frac{P_i - \sqrt{D}}{Q_i} \right) (Q_i, P_i) . \quad (3.5)$$

Additionally, we can generalize [JSW, Lemma 6.3] and define

$$\begin{aligned} F &= aP_0 + cR_0 , & G &= aQ_0 + cP_0 , \\ F' &= bP_0 + dR_0 , & G' &= bQ_0 + dP_0 . \end{aligned}$$

Then,

$$\begin{aligned}
 Q_i &= \varepsilon(a^2Q_0 + 2acP_0 + c^2R_0) & P_i &= \varepsilon(abQ_0 + (ad + bc)P_0 + cdR_0) & (3.6) \\
 &= \varepsilon(aG + cF) , & &= \varepsilon(bG + dF) , \\
 R_i &= \varepsilon(b^2Q_0 + 2bdP_0 + d^2R_0) \\
 &= \varepsilon(bG' + dF') .
 \end{aligned}$$

Q_i and P_i can be given expressly in terms of Q_0 and P_0 by making the appropriate substitution for R_0 :

$$\begin{aligned}
 Q_i &= \varepsilon \left(a^2Q_0 + 2acP_0 + c^2 \frac{P_0^2 - D}{Q_0} \right) \\
 &= \varepsilon \frac{a^2Q_0^2 + 2acQP_0 + c^2P_0^2 - c^2D}{Q_0} \\
 &= \varepsilon \frac{G^2 - c^2D}{Q_0} , & (3.7)
 \end{aligned}$$

$$\begin{aligned}
 P_i &= \varepsilon \left(abQ_0 + (ad + bc)P_0 + cd \frac{P_0^2 - D}{Q_0} \right) \\
 &= \varepsilon \frac{abQ_0^2 + (ad + bc)QP_0 + cdP_0^2 - cdD}{Q_0} \\
 &= \varepsilon \frac{GG' - cdD}{Q_0} . & (3.8)
 \end{aligned}$$

Alternatively, if Q_i has been computed, a computationally more efficient formula for P_i is

$$\begin{aligned}
 P_i &= \varepsilon(bG + dF) = \varepsilon \left(\frac{ad - \varepsilon}{c} G + dF \right) \\
 &= \varepsilon \frac{d(aG + cF) - \varepsilon G}{c} = \frac{d\varepsilon(aG + cF) - \varepsilon^2 G}{c}
 \end{aligned}$$

$$= \frac{dQ_i - G}{c} . \quad (3.9)$$

In subsequent chapters, these formulas will allow us to perform many reduction steps without needing to calculate Q_i , P_i , and R_i at each step.

3.3 Reduction

3.3.1 Purpose of Reduction

It is clear from the multiplication formulas of Section 3.1 that the magnitude of the parameters Q_0 and P_0 of the product ideal is $O(Q_a Q_b)$. Theoretically this does not pose a problem; however, it is not practical to compute with numbers that grow this fast. Informally, reduction is the process of finding an ideal with parameters bounded in magnitude by \sqrt{D} that is equivalent to an ideal with larger parameters.

Definition 3.6. An ideal $\mathfrak{a} = (Q, P)$ of \mathcal{O} is reduced if \mathfrak{a} is primitive and there does not exist an element $\beta \neq 0 \in \mathfrak{a}$ such that $|\beta| < N(\mathfrak{a})$ and $|\bar{\beta}| < N(\mathfrak{a})$. For ideals of positive discriminant where $\alpha = (P + \sqrt{D})/Q$, this is equivalent to $\alpha > 1$ and $-1 < \bar{\alpha} < 0$. This is the case if $0 < P < \sqrt{D}$, $0 < Q < P + \sqrt{D}$ and $Q + P > \sqrt{D}$. Ideals of negative discriminant are reduced if $|2P| < Q < R$ or $0 \leq 2P \leq Q \leq R$.

3.3.2 Classical Reduction of Ideals (Lagrange)

Lagrange reduction of an ideal (Q_0, P_0) of positive discriminant is the process of developing the simple continued fraction expansion of $(P_0 + \sqrt{D})/Q_0$. To develop the simple continued fraction expansion of a real number α , we let $\alpha_0 = \alpha$, $q_i = \lfloor \alpha_i \rfloor$,

and $\alpha_{i+1} = 1/(\alpha_i - q_i)$. Note that $\alpha_{i+1} > 1$ for all $i \geq 0$. Using these definitions, α may be written as

$$\alpha = q_0 + \frac{1}{q_1 + \frac{1}{q_2 + \frac{1}{\ddots q_{n-1} + \frac{1}{\alpha_n}}}} .$$

We write the above compactly as

$$\alpha = [q_0, q_1, q_2, \dots, q_{n-1}, \alpha_n] .$$

The variables q_i are termed *partial quotients*. If we define the sequences

$$\begin{aligned} A_{-2} &= 0, & A_{-1} &= 1, & A_i &= q_i A_{i-1} + A_{i-2} , \\ B_{-2} &= 1, & B_{-1} &= 0, & B_i &= q_i B_{i-1} + B_{i-2} , \end{aligned} \tag{3.10}$$

then the rational numbers $A_i/B_i = [q_0, q_1, \dots, q_i]$ are termed the *convergents* of the simple continued fraction expansion of α . If α is rational then $\alpha = [q_0, q_1, \dots, q_n]$ for some $n \in \mathbb{N}$. If α is irrational, then we have an infinite continued fraction and $\lim_{i \rightarrow \infty} A_i/B_i = \alpha$.

With this notation, the sequence of ideals $\{(Q_i, P_i)\}$ is generated by

$$\frac{P_0 + \sqrt{D}}{Q_0} = \left[q_0, q_1, \dots, q_{i-1}, \frac{P_i + \sqrt{D}}{Q_i} \right]$$

where the values q_i, Q_i, P_i are obtained from Algorithm 3.1 below.

Algorithm 3.1 $\text{Rho}(\rho)$ — one continued fraction expansion step

Input: $(Q_i, P_i), D$

Output: $\rho(Q_i, P_i) = (Q_{i+1}, P_{i+1})$

$$\begin{aligned} q_i &\leftarrow \left\lfloor \frac{P_i + \sqrt{D}}{Q_i} \right\rfloor \\ P_{i+1} &\leftarrow q_i Q_i - P_i \\ Q_{i+1} &\leftarrow \frac{D - P_{i+1}^2}{Q_i} \end{aligned}$$

return (Q_{i+1}, P_{i+1})

By the previous discussion of continued fractions, $\alpha_i = \frac{P_i + \sqrt{D}}{Q_i} > 1$ for $i > 0$. This is the first condition of the reduction criterion and guarantees that $Q_i > 0$ if $|P_i| < \sqrt{D}$, and P_i and Q_i have the same sign if $|P_i| > \sqrt{D}$. From the latter fact we may conclude that if $Q_i > 0$ and $P_i < \sqrt{D}$, then $|P_i| < \sqrt{D}$. Under these conditions, $\bar{\alpha} < 0$; hence, by Theorem 4.2 of [WW87], (Q_i, P_i) is reduced. This provides a simplified reduction criterion for use with this method of reduction. We emphasize that this criterion may only be used if $\alpha_i > 1$ which is only generally guaranteed for $i > 0$. One further observation we may make from the formula for Q_{i+1} is that the sign of Q_{i+1} is the opposite of Q_i when $|P_{i+1}| > \sqrt{D}$ and remains the same if $|P_{i+1}| < \sqrt{D}$.

We now illustrate how to use the ideas of Sections 3.2.1 and 3.2.2 to create an algorithm that implements Lagrange reduction and calculates a relative generator. First, we derive a theoretical algorithm using matrices and use it to develop a theoretical algorithm using ideals and a practical algorithm which is easily implemented.

To find the matrix that effects Lagrange reduction, we need to solve for N and ε

in Definition 3.4. A little arithmetic yields the matrix $\begin{pmatrix} q_i & -1 \\ -1 & 0 \end{pmatrix}$ and $\varepsilon = -1$. Hence, we have Algorithm 3.2 which effects a full reduction using matrices.

Algorithm 3.2 Continued Fraction Reduction (Matrix)

Input: $\mathfrak{A}_0 = \begin{pmatrix} Q_0 & P_0 \\ P_0 & R_0 \end{pmatrix}$, D

Output: \mathfrak{A}_i , N_i such that $\mathfrak{A}_i = \phi_{N_i}(\mathfrak{A}_0)$ and (Q_i, P_i) is reduced

[Initialization]

$i \leftarrow 0$

$N_0 \leftarrow I_2$

[Reduction]

while (Q_i, P_i) is not reduced **do**

$$q_i \leftarrow \left\lfloor \frac{P_i + \sqrt{D}}{Q_i} \right\rfloor$$

$$\begin{pmatrix} Q_{i+1} & P_{i+1} \\ P_{i+1} & R_{i+1} \end{pmatrix} = (-1) \begin{pmatrix} q_i & -1 \\ -1 & 0 \end{pmatrix}^T \begin{pmatrix} Q_i & P_i \\ P_i & R_i \end{pmatrix} \begin{pmatrix} q_i & -1 \\ -1 & 0 \end{pmatrix}$$

$$N_{i+1} \leftarrow N_i \begin{pmatrix} q_i & -1 \\ -1 & 0 \end{pmatrix}$$

$i \leftarrow i + 1$

return \mathfrak{A}_i , N_i

Now we will translate this into an algorithm which uses only ideals. We want to determine the relative generator at each step in the algorithm so that $(\psi_{i+1})\mathfrak{a}_i = \mathfrak{a}_{i+1}$. The relative generator corresponding to no reduction step is $\psi_0 = 1$; we can confirm this using (3.4) and the identity matrix in which case $a = d = 1$, $b = c = 0$ and $\varepsilon = 1$. The relative generator for every other step corresponds to the matrix $\begin{pmatrix} q_i & -1 \\ -1 & 0 \end{pmatrix}$ in which case $a = q_i$, $b = c = -1$, $d = 0$ and $\varepsilon = -1$. Plugging these values into (3.4) yields $\psi_{i+1} = -(q_i - \bar{\alpha}_i)$. These calculations give Algorithm 3.3.

Now to create an algorithm that is useful in practice, we can show that in Algo-

Algorithm 3.3 Continued Fraction Reduction (Ideal)

Input: $\alpha_0 = (Q_0, P_0)$ **Output:** α_i, Ψ_i , such that $\alpha_i = (\Psi_i)\alpha_0$ and α_i is reduced

[Initialization]

 $i \leftarrow 0$ $\psi_0 \leftarrow 1$

[Reduction]

while α_i is not reduced **do** $q_i \leftarrow \lfloor \alpha_i \rfloor$ $\psi_{i+1} \leftarrow \bar{\alpha}_i - q_i$ $\alpha_{i+1} \leftarrow (\psi_{i+1})\alpha_i$ $i \leftarrow i + 1$

[Compute relative generator]

 $\Psi_i \leftarrow \prod_{j=0}^i \psi_j$ **return** α_i, Ψ_i

rithm 3.2 the matrices N_i and $\det(N_i)$ are given by

$$N_i = \begin{pmatrix} A_{i-1} & -A_{i-2} \\ -B_{i-1} & B_{i-2} \end{pmatrix}, \quad (3.11)$$

$$\det(N_i) = A_{i-1}B_{i-2} - A_{i-2}B_{i-1} = (-1)^i. \quad (3.12)$$

Hence, $\Psi_i = (-1)^i(A_{i-1} - B_{i-1}\bar{\alpha}_0) = (B_{i-2} + B_{i-1}\bar{\alpha}_i)^{-1}$ by Equations (3.4) and (3.5).

To find the values for Q_{i+1}, P_{i+1} and R_{i+1} we compute

$$\begin{pmatrix} Q_{i+1} & P_{i+1} \\ P_{i+1} & R_{i+1} \end{pmatrix} = - \begin{pmatrix} q_i & -1 \\ -1 & 0 \end{pmatrix}^T \begin{pmatrix} Q_i & P_i \\ P_i & R_i \end{pmatrix} \begin{pmatrix} q_i & -1 \\ -1 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} -q_i(q_i Q_i - P_i) + q_i P_i - R_i & q_i Q_i - P_i \\ q_i Q_i - P_i & -Q_i \end{pmatrix}. \quad (3.13)$$

We are now in a position to present Algorithm 3.4. The computations for Q_{i+1} and P_{i+1} utilize some efficiencies discovered by Tenner. The justification for the computation of Q_{i+1} can be seen by inspection of the matrix of Equation (3.13). The formula for P_{i+1} is obtained by solving $r_i = P_i + \lfloor \sqrt{D} \rfloor - q_i Q_i$ (the remainder of $\frac{P_i + \lfloor \sqrt{D} \rfloor}{Q_i}$) for $q_i Q_i - P_i$.

Algorithm 3.4 Continued Fraction Reduction (Implementation)

Input: $\alpha_0 = (Q_0, P_0)$, D

Output: $\alpha_i = (Q_i, P_i)$, Ψ_i , such that $\alpha_i = (\Psi_i)\alpha_0$ and α_i is reduced

Note: The $\{A_i\}$ do not need to be calculated unless they are required by an external algorithm.

[Initialization]

$i \leftarrow 0$

$A_{-2} \leftarrow 0, A_{-1} \leftarrow 1, B_{-2} \leftarrow 1, B_{-1} \leftarrow 0$

$R_0 \leftarrow \frac{P_0^2 - D}{Q_0}$

[Reduction]

while (Q_i, P_i) is not reduced **do**

$q_i \leftarrow \left\lfloor \frac{P_i + \lfloor \sqrt{D} \rfloor}{Q_i} \right\rfloor, r_i \leftarrow P_i + \lfloor \sqrt{D} \rfloor - q_i Q_i$ (simultaneously)

$R_{i+1} \leftarrow -Q_i$

$P_{i+1} \leftarrow \lfloor \sqrt{D} \rfloor - r_i$

$Q_{i+1} \leftarrow q_i(P_i - P_{i+1}) - R_i$

$A_i \leftarrow q_i A_{i-1} + A_{i-2}, B_i \leftarrow q_i B_{i-1} + B_{i-2}$

$i \leftarrow i + 1$

[Compute relative generator]

$\Psi_i^{-1} \leftarrow B_{i-2} + B_{i-1} \left(\frac{P_i - \sqrt{D}}{Q_i} \right)$

return $(Q_i, P_i), \Psi_i$

We will see in Chapter 5 that many of the faster reduction algorithms do not completely reduce the ideal but come within a few Lagrange steps of being reduced. Therefore, the Lagrange algorithm is often used after the completion of other reduction algorithms to finish the reduction. The following lemma is an important technical result that simplifies the computation of the relative generator in these cases.

Lemma 3.7. *If $(Q, P) = \left(d - c\left(\frac{P' - \sqrt{D}}{Q'}\right)\right) (Q', P')$, then calling Algorithm 3.4 with (Q', P') and initializing $B_{-2} \leftarrow d$, $B_{-1} \leftarrow -c$ produces (Q'', P'') and Ψ such that $(Q'', P'') = (\Psi)(Q, P)$.*

Proof. Let $N' = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ be such that $\mathfrak{A}' = \phi_{N'}(\mathfrak{A})$ and let $\mathfrak{A}'' = \phi_{N''}(\mathfrak{A}')$ where N'' is returned from Algorithm 3.2 on input of \mathfrak{A}' . This implies that

$$\begin{aligned} N'N'' &= \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \prod_{j=0}^{i-1} \begin{pmatrix} q_j & -1 \\ -1 & 0 \end{pmatrix} && \text{(for some } i\text{)} \\ &= \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} A_{-1} & -A_{-2} \\ -B_{-1} & B_{-2} \end{pmatrix} \prod_{j=0}^{i-1} \begin{pmatrix} q_j & -1 \\ -1 & 0 \end{pmatrix} && \text{(by (3.10) and (3.11))} . \end{aligned}$$

Hence, if in Algorithm 3.2 we set

$$N_0 = \begin{pmatrix} A_{-1} & -A_{-2} \\ -B_{-1} & B_{-2} \end{pmatrix} \leftarrow \begin{pmatrix} a & b \\ c & d \end{pmatrix} \text{ instead of } \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} ,$$

the matrix N_i returned is $N'N''$ from which we may calculate the relative generator such that $(\Psi)(Q'', P'') = (Q, P)$. In the case of Algorithm 3.4 we do not need the $\{A_i\}$ sequence, only the $\{B_i\}$ sequence. By Equation (3.5), the values c and d from $(Q, P) = \left(d - c\left(\frac{P' - \sqrt{D}}{Q'}\right)\right) (Q', P')$ are obtained from the bottom row of the matrix

N' ; therefore, all that is required is to set $B_{-2} \leftarrow d$ and $B_{-1} \leftarrow -c$ in which case the relative generator returned satisfies $(Q'', P'') = (\Psi)(Q, P)$. \square

3.4 Finding a Reduced Representative

In the case of imaginary quadratic fields, it is well known that for any ideal \mathfrak{a}_0 , there is a unique reduced ideal \mathfrak{a}_r equivalent to it. For real quadratic fields, we know that there can be many equivalent reduced ideals. If the ρ -operation of Algorithm 3.1 is applied to a reduced ideal \mathfrak{a}_r , then $\rho^n(\mathfrak{a}_r)$ is reduced for any $n \geq 0$ (see, for example, [WW87]). The ideals produced are ultimately periodic and so form a cycle of reduced ideals; in fact, this cycle contains *all* of the reduced ideals equivalent to \mathfrak{a}_r . Each equivalence class in the class group contains exactly one cycle and so the number of distinct cycles of \mathcal{O} is equal to the class number. If α_r is the quadratic irrational corresponding to \mathfrak{a}_r , then one may move in the reverse direction of ρ in the cycle by developing the simple continued fraction expansion of $-\bar{\alpha}_r$; this operation is denoted ρ^{-1} and $\rho\rho^{-1}(\mathfrak{a}_r) = \rho^{-1}\rho(\mathfrak{a}_r) = \mathfrak{a}_r$, if \mathfrak{a}_r is reduced.¹

If we use Algorithm 3.3 to reduce an ideal \mathfrak{a}_0 to \mathfrak{a}_r with r minimal, we know by [vtW01, Cor. 4.6] that if $(\Psi_r)\mathfrak{a}_0 = \mathfrak{a}_r$, then $|\Psi_r| \leq 1$. We can pick a unique representative from the cycle in the following manner. Let $\theta_i = A_i - B_i\bar{\alpha}_r$ (see Equation (3.4)) be the relative generator of two reduced ideals \mathfrak{a}_r and \mathfrak{a}_{r+i} in the cycle of reduced ideals; that is, $(\theta_i)\mathfrak{a}_r = \mathfrak{a}_{r+i}$. Applying Equation (3.4), it is easy

¹Concepts we won't require but are related are that if \mathfrak{a} is reduced, $n > 0$ is minimal such that $\mathfrak{a} = \rho^n(\mathfrak{a})$, and Ψ_n is the associated relative generator, then Ψ_n is the *fundamental unit* and $\ln \Psi_n$ is the *regulator*. If \mathfrak{a} and \mathfrak{b} are equivalent reduced ideals such that $\mathfrak{a} = (\Psi)\mathfrak{b}$, then $\ln |\Psi|$ modulo the regulator is the *Shanks distance*, from \mathfrak{a} to \mathfrak{b} . These concepts were further developed by Shanks and termed the *infrastructure of the class group* by him. For further details, see [Sha72], [Len82] and [Sch82].

to show that $1 = \theta_0 < \theta_1 < \theta_2 < \dots < \theta_i$ for all i . Hence, for some i we obtain $(\theta_i \Psi_r) \mathfrak{a}_0 = \mathfrak{a}_{r+i}$ where $|\theta_i \Psi_r| \leq 1 < |\theta_{i+1} \Psi_r|$. The ideal \mathfrak{a}_{r+i} is referred to as the *near reduced* ideal of \mathfrak{a}_0 . Note that the term “near reduced” does not imply that the ideal is not fully reduced. The idea is that \mathfrak{a}_0 and \mathfrak{a}_{r+i} are near to each other since the relative generator is close to 1.

Algorithm 3.5 is a theoretical algorithm illustrating the procedure to obtain a near reduced ideal. For a practical implementation, see [JSW].

Algorithm 3.5 Near Reduced Ideal (Ideal)

Input: A reduced ideal \mathfrak{a}_r and irrational number Ψ_r where $|\Psi_r| \leq 1$ and $(\Psi_r) \mathfrak{a}_0 = \mathfrak{a}_r$
Output: A near reduced ideal \mathfrak{a}_{r+i} and number θ_i such that $(\theta_i \Psi_r) \mathfrak{a}_0 = \mathfrak{a}_{r+i}$

```

[ Initialization ]
 $i \leftarrow -1$ 
 $\theta_0 \leftarrow 1$ 

[ Compute near reduced ideal ]
while  $|\theta_{i+1} \Psi_r| \leq 1$  do
     $i \leftarrow i + 1$ 
     $q_i \leftarrow \lfloor \alpha_{r+i} \rfloor$ 
     $\theta_{i+1} \leftarrow \theta_i (\bar{\alpha}_{r+i} - q_i)$ 
     $\mathfrak{a}_{r+i+1} \leftarrow (\theta_{i+1}) \mathfrak{a}_r$ 
return  $\mathfrak{a}_{r+i}, \theta_i$ 

```

3.5 Binary Quadratic Forms

Ideals of quadratic fields and binary quadratic forms are closely related objects. Most of the algorithms presented in this thesis were originally presented in the language of binary quadratic forms. An in-depth discussion and proof of the equivalence of ideals and classes of forms is readily available in works such as [Bue89] or [Coh93]. In this

section we show how to convert the algorithms for forms into algorithms for ideals and present the classical Gaussian reduction of forms.

Definition 3.8. A *binary quadratic form* is a function $f(x, y) = ax^2 + bxy + cy^2$ with $a, b, c, x, y \in \mathbb{Z}$. It is written compactly as (a, b, c) and its *discriminant* is defined to be $\Delta = b^2 - 4ac$. Note that $\Delta \equiv 0, 1 \pmod{4}$.

Definition 3.9. If $\gcd(a, b, c) = 1$ then (a, b, c) is a *primitive* quadratic form. The form (a, b, c) is *definite* if $\Delta < 0$ and *indefinite* if $\Delta > 0$.

The above references ([Bue89], [Coh93]) show that a form (a, b, c) can be put in correspondence with the \mathbb{Z} -module

$$a\mathbb{Z} \oplus \left(\frac{-b + \sqrt{\Delta}}{2} \right) \mathbb{Z} .$$

This is not a one-to-one mapping as many distinct forms correspond to a single \mathbb{Z} -module. To convert algorithms for working with forms to those for ideals, we use the \mathbb{Z} -module from the above correspondence and set it equal to an arbitrary \mathbb{Z} -module that uses our representation of ideals.

$$a\mathbb{Z} \oplus \frac{-b + \sqrt{\Delta}}{2} \mathbb{Z} = \frac{Q}{\sigma} \mathbb{Z} \oplus \frac{P + \sqrt{D}}{\sigma} \mathbb{Z}$$

Hence, one correspondence that will work is

$$a = \frac{Q}{\sigma} \text{ and } \frac{-b + \sqrt{\Delta}}{2} = \frac{P + \sqrt{D}}{\sigma} .$$

Equating the rational parts of the right hand side equation,

$$b = -\frac{2P}{\sigma}.$$

We do not have (nor do we require) a one-to-one mapping between ideals and forms but this does give us a correspondence so that we may translate the algorithms. In fact, for the algorithms, we may further simplify the correspondence to $a = Q/\sigma$ and $b = 2P/\sigma$. A few more calculations confirm that $c = R/\sigma$.

Gaussian reduction is the standard method used to reduce forms. We first outline the reduction method and then translate it for use with ideals. Later, we will compare this method with Lagrange's method.

Definition 3.10. Two forms f and g are *properly equivalent* (denoted $f \sim g$) if there exists an integral 2×2 matrix $\begin{pmatrix} e & f \\ g & h \end{pmatrix}$ with determinant 1 (a *unimodular* 2×2 matrix) such that $g(x, y) = f(x', y')$ where $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} e & f \\ g & h \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$. The forms are *improperly equivalent* if the matrix has determinant -1 .

Forms with negative discriminant are reduced if $|b| < a < c$ or $0 \leq b \leq a \leq c$ and forms with positive discriminant are reduced if $|\sqrt{\Delta} - 2|a|| < b < \sqrt{\Delta}$. For any arbitrary form f , traditional Gaussian reduction reduces a form by applying the unimodular transformations

$$N = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \text{ or } N = \begin{pmatrix} 1 & q \\ 0 & 1 \end{pmatrix}$$

where the computation of q will be given later in this section. Using the matrix

$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$, we obtain

$$\begin{aligned} \begin{pmatrix} x' \\ y' \end{pmatrix} &= \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} y \\ -x \end{pmatrix} \\ \Rightarrow (a, b, c) &= ay^2 + b(-x)y + c(-x)^2 \\ \Rightarrow (a, b, c) &\sim (c, -b, a) . \end{aligned}$$

The matrix $\begin{pmatrix} 1 & q \\ 0 & 1 \end{pmatrix}$ gives

$$\begin{aligned} \begin{pmatrix} x' \\ y' \end{pmatrix} &= \begin{pmatrix} 1 & q \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x + qy \\ y \end{pmatrix} \\ \Rightarrow (a, b, c) &= a(x + qy)^2 + b(x + qy)(y) + cy^2 \\ &= ax^2 + (2aq + b)xy + (aq^2 + bq + c)y^2 \\ \Rightarrow (a, b, c) &\sim (a, 2aq + b, aq^2 + bq + c) . \end{aligned}$$

Rather than working through the details of a variable substitution, an alternative method to determine how N affects the form is to write the form (a, b, c) as $\begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix}$, then $ax^2 + bxy + cy^2 = \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$. To transform $\begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix}$ we now compute

$N^T \begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix} N$. With the matrix $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ we obtain

$$\begin{aligned}
 & \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \\
 &= \begin{pmatrix} -b/2 & -c \\ a & b/2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \\
 &= \begin{pmatrix} c & -b/2 \\ -b/2 & a \end{pmatrix} \\
 &\Rightarrow (a, b, c) \sim (c, -b, a)
 \end{aligned}$$

and the matrix $\begin{pmatrix} 1 & q \\ 0 & 1 \end{pmatrix}$ yields

$$\begin{aligned}
 & \begin{pmatrix} 1 & 0 \\ q & 1 \end{pmatrix} \begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix} \begin{pmatrix} 1 & q \\ 0 & 1 \end{pmatrix} \\
 &= \begin{pmatrix} a & b/2 \\ aq + b/2 & bq/2 + c \end{pmatrix} \begin{pmatrix} 1 & q \\ 0 & 1 \end{pmatrix} \\
 &= \begin{pmatrix} a & aq + b/2 \\ aq + b/2 & aq^2 + bq + c \end{pmatrix} \\
 &\Rightarrow (a, b, c) \sim (a, 2aq + b, aq^2 + bq + c)
 \end{aligned}$$

as before.

Hence, to reduce a form using these transformations the procedure is to make

$|a| \leq |c|$ using the first transformation and minimize the size of b using the second transformation. To find the appropriate value for q , we use Euclidean division to find q such that $b = -2aq + r$ with $-a < r \leq a$.

Note that various steps may be combined into one. For example, if we apply the first transformation, followed by the second we obtain

$$(a, b, c) \sim (c, 2cq - b, a - bq + cq^2) . \quad (3.14)$$

Readers familiar with forms will have noticed that we have allowed matrices of negative determinant in the reductions of Section 3.3.2. When computing with forms, such transformations are often not allowed since the resulting form is not necessarily properly equivalent to the initial form. The Lagrange reduction technique would need to be modified slightly to be used with forms but ideals may use standard Gaussian reduction. This is given in terms of matrices as Algorithm 3.6 and in terms of ideals as Algorithm 3.7.

To produce an algorithm that is useful in practice, we use the same procedure utilized in the previous section. In this case we arrive at the sequences

$$\begin{aligned} X_{-2} &= 1, & X_{-1} &= 0, & X_i &= q_i X_{i-1} - X_{i-2} \\ Y_{-2} &= 0, & Y_{-1} &= 1, & Y_i &= q_i Y_{i-1} - Y_{i-2} \end{aligned}$$

and obtain the matrix

$$N_i = \begin{pmatrix} X_{i-2} & X_{i-1} \\ Y_{i-2} & Y_{i-1} \end{pmatrix} .$$

Algorithm 3.6 Gaussian Reduction (Matrix)

Input: $\mathfrak{A}_0 = \begin{pmatrix} Q_0 & P_0 \\ P_0 & R_0 \end{pmatrix}, D$ **Output:** \mathfrak{A}_i, N_i such that $\mathfrak{A}_i = \phi_{N_i}(\mathfrak{A}_0)$ and (Q_i, P_i) is reduced

[Initialization]

 $i \leftarrow 0$ $N_0 \leftarrow I_2$

[Reduction]

while (Q_i, P_i) is not reduced **do**

$$q_i \leftarrow \frac{R_i}{|R_i|} \left\lfloor \frac{P_i + \sqrt{D}}{|R_i|} \right\rfloor$$

$$\begin{pmatrix} Q_{i+1} & P_{i+1} \\ P_{i+1} & R_{i+1} \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & q_i \end{pmatrix}^T \begin{pmatrix} Q_i & P_i \\ P_i & R_i \end{pmatrix} \begin{pmatrix} 0 & -1 \\ 1 & q_i \end{pmatrix}$$

$$N_{i+1} \leftarrow N_i \begin{pmatrix} 0 & -1 \\ 1 & q_i \end{pmatrix}$$

 $i \leftarrow i + 1$ **return** \mathfrak{A}_i, N_i

Note that $\det(N_i) = 1$ for all i since $\det \begin{pmatrix} 0 & -1 \\ 1 & q_i \end{pmatrix} = 1$; hence, Equations (3.4) and (3.5) give $\Psi_i = X_{i-2} + Y_{i-2}\bar{\alpha}_0 = (Y_{i-1} - Y_{i-2}\bar{\alpha}_i)^{-1}$. These algorithms assume $D > 0$; if $D < 0$, the only change is that $q_i \leftarrow \lfloor P_i/R_i \rfloor$.

In [KW90], Kaplan and Williams give another version of Gaussian reduction. Their version is identical to the Lagrange reduction of Section 3.3.2 except that the value of q_i is given by

$$q_i = \frac{Q_i}{|Q_i|} \left\lfloor \frac{P_i + \sqrt{D}}{|Q_i|} \right\rfloor.$$

Algorithm 3.7 Gaussian Reduction (Ideal)

Input: $\mathfrak{a}_0 = (Q_0, P_0), D$ **Output:** \mathfrak{a}_i, Ψ_i such that $\mathfrak{a}_i = (\Psi_i)\mathfrak{a}_0$ and \mathfrak{a}_i is reduced

[Initialization]

 $i \leftarrow 0$ $\psi_0 \leftarrow 1$ $R_0 \leftarrow \frac{P_0^2 - D}{Q_0}$

[Reduction]

while \mathfrak{a}_i is not reduced **do** $q_i \leftarrow \frac{R_i}{|R_i|} \left\lfloor \frac{P_i + \sqrt{D}}{|R_i|} \right\rfloor$ $\psi_{i+1} \leftarrow \bar{\alpha}_i$ $\mathfrak{a}_{i+1} \leftarrow (\psi_{i+1})\mathfrak{a}_i$ $i \leftarrow i + 1$

[Compute relative generator]

 $\Psi_i \leftarrow \prod_{j=0}^i \psi_j$ **return** \mathfrak{a}_i, Ψ_i

They prove that Lagrange reduction terminates in at most

$$\max \left(\frac{\log(|Q_0|/2\sqrt{D})}{2\log(1+\sqrt{5})/2} + \frac{5}{2}, 2 \right)$$

reduction steps while their Gaussian reduction terminates in at most

$$\max \left(\frac{|Q_0|}{2\sqrt{D}} + 1, 2 \right)$$

steps. Buchmann proves in [Buc03, Thm. 6.5.3] that an ideal is reduced by the

algorithm presented here in at most

$$\frac{1}{2} \log \left(\frac{|Q_0|}{2\sqrt{D}} \right) + 2$$

steps.

Algorithm 3.8 Gaussian Reduction (Implementation)

Input: $\mathfrak{a}_0 = (Q_0, P_0)$, D

Output: $\mathfrak{a}_i = (Q_i, P_i)$, Ψ_i , such that $\mathfrak{a}_i = (\Psi_i)\mathfrak{a}_0$ and \mathfrak{a}_i is reduced

[Initialization]

$i \leftarrow 0$

$Y_{-2} \leftarrow 0, Y_{-1} \leftarrow 1$

$R_0 \leftarrow \frac{P_0^2 - D}{Q_0}$

[Reduction]

while (Q_i, P_i) is not reduced **do**

$q_i \leftarrow \left\lfloor \frac{P_i + \lfloor \sqrt{D} \rfloor}{|R_i|} \right\rfloor, r_i \leftarrow P_i + \lfloor \sqrt{D} \rfloor - q_i |R_i|$ (simultaneously)

$q_i \leftarrow \text{sign}(R_i)q_i$

$Q_{i+1} \leftarrow R_i$

$P_{i+1} \leftarrow \lfloor \sqrt{D} \rfloor - r_i$

$R_{i+1} \leftarrow Q_i + q_i(P_{i+1} - P_i)$

$Y_i \leftarrow q_i Y_{i-1} - Y_{i-2}$

$i \leftarrow i + 1$

[Compute relative generator]

$\Psi_i^{-1} \leftarrow Y_{i-1} - Y_{i-2} \left(\frac{P_i - \sqrt{D}}{Q_i} \right)$

return (Q_i, P_i) , Ψ_i

Chapter 4

Applications to Secure Communication

One application of ideals is the exchange of a secret key by two parties across public communication channels. In this chapter, core concepts of secure communication are explained and a key agreement protocol utilizing ideals of real quadratic fields is outlined. In the process of exchanging keys, thousands of reductions are needed; accordingly, it provides an excellent system in which to benchmark the reduction algorithms as well as demonstrate the important role they play in a real-life application.

For two parties to communicate securely across publicly accessible channels, they must first agree upon the cryptosystem that they will use as well as a key that will enable their communication to remain private. By convention, we refer to the two communicating parties as Alice and Bob. To use symmetric-key cryptography, Alice and Bob require a common key. This key must somehow be securely communicated between the two of them. The security of the delivery method depends upon their

security requirements. It may take the form of delivery in person, by trusted courier or perhaps an electronic channel that they know is sufficiently secure during the transmission of the key. After the key has been exchanged, Alice then encrypts her message and the ciphertext is sent across insecure public channels where we assume there are eavesdroppers who may obtain the ciphertext. This information is reflected in Figure 4.1. If the parties are across town, a key may be delivered in person; however, should one wish to communicate with someone on another continent he has never met, the problem becomes much more difficult.

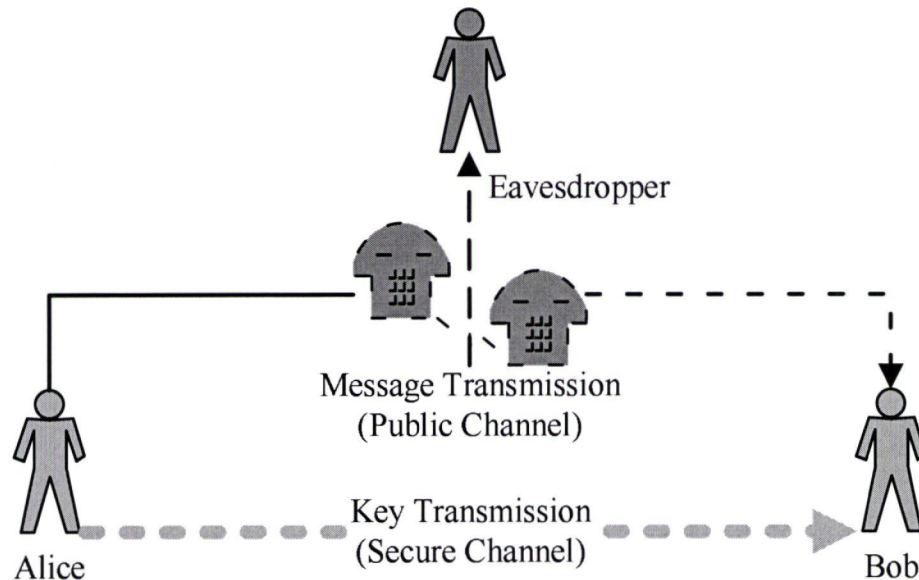


Figure 4.1: Message And Key Transmission

We refer to this as the *key distribution problem* and it was this obstacle which was a road block for the proliferation of secure communications. In their seminal paper [DH76], Whitfield Diffie and Martin Hellman revolutionized the science of cryptography by providing a method of exchanging keys over insecure public channels,

completely obviating the need for a secure channel. Their idea was to use a *one-way function*, which can informally be defined as a function f such that if we are given x in the domain of f , $f(x)$ is easy to compute, but given y in the range of f , finding x such that $f(x) = y$ is computationally difficult.

In their paper, the one-way function they proposed for key exchange was exponentiation in a finite cyclic group. Reversing this operation is known as the *Discrete Logarithm Problem* (DLP). The problem is, given a finite cyclic group¹ G generated by g , to quickly compute (deterministically in polynomial time) the integer x (with $0 \leq x < |G|$ where $|G|$ denotes the number of elements of G) given the element g^x . A proof that this problem is a one-way function would require a proof that there does not exist a deterministic polynomial time algorithm that computes x . Since this problem can clearly be solved nondeterministically in polynomial time, a proof that a deterministic algorithm does not exist would also prove the well-known complexity theory conjecture that $\mathcal{P} \neq \mathcal{NP}$.² Because the DLP (nor any other problem) has not been proved to be a one-way function, the most we can say is that it has resisted the intensive efforts of many gifted mathematicians and only modest progress has been made toward a solution.

Should Alice and Bob wish to exchange a key, the protocol is:

1. Agree on a finite cyclic group G and a generator g . (*public*)
2. Alice picks a random integer x with $0 < x < |G|$. (*secret*)
3. Bob picks a random integer y with $0 < y < |G|$. (*secret*)

¹In fact, all that is required is a finite semi-group.

² \mathcal{P} is the set of decision problems that can be solved deterministically in polynomial time and \mathcal{NP} is the set of decision problems that can be solved nondeterministically in polynomial time.

4. Alice computes $a = g^x$, and sends a to Bob. (*public*)
5. Bob computes $b = g^y$, and sends b to Alice. (*public*)
6. Alice computes $k = b^x$. (*secret*)
7. Bob computes $k' = a^y$. (*secret*)

Since $k = b^x = (g^y)^x = g^{yx} = g^{xy} = (g^x)^y = a^y = k'$, they now share a common key. Note that an eavesdropper has all of the information required to compute the key, but since finding x given g^x is computationally infeasible if $|G|$ is large, the key may be considered secret.³

The focus of this thesis is on real quadratic fields but we mention that an imaginary quadratic field based key exchange was introduced by Buchmann and Williams in [BW88a]. This cryptosystem is based on a finite group, namely the class group. Here, Alice and Bob agree on a class group, which is specified by the imaginary quadratic field $\mathbb{Q}(\sqrt{D})$, and an element g of the group (an equivalence class). While each equivalence class contains an infinite number of elements, a unique reduced ideal exists in each class which may be considered the class representative. They exponentiate g as per the Diffie-Hellman protocol, and the key is based upon the reduced result. Imaginary quadratic field cryptography has been extensively studied by Buchmann and Hamdy in [BH01].

The majority of today's popular cryptosystems rely on the difficulty of factoring or solving the DLP in a finite group. In [BW90], Buchmann and Williams introduced

³ It should be mentioned that the Diffie-Hellman protocol might not actually require the DLP to be solved in order to break the system. If a fast method is found that can find g^{xy} given g^x and g^y then the system is broken. This is referred to as the Diffie-Hellman problem and it is not known whether the Diffie-Hellman problem and DLP are equivalent. Certainly solving the DLP will break the Diffie-Hellman key exchange, but the converse might not be true.

the first Diffie-Hellman type public-key exchange system not based on a finite group. Instead, it is based on the set of reduced principal ideals of a real quadratic field. We have closure on this set under the operation of multiplication followed by reduction; however, under this operation we lose associativity and hence we do not have a group. To pick a reduced representative from the set, we use Algorithm 3.5. Since this key exchange is not based on a group structure, it may possibly remain secure even if the DLP for finite groups is solved.

The following is a theoretical key-exchange protocol utilizing ideals of the ring of integers of a real quadratic field. In practice, an exact representation of the relative generators cannot be sent because the size of the parameters defining them is exponential in D . Hence, an approximation such as the (f, p) representation of [JSW] must be used instead. This is explained further in Chapter 6.

1. Agree on a real quadratic field $K = \mathbb{Q}(\sqrt{D})$ where $D > 0$ is square-free and a reduced principal ideal $\mathfrak{g} \in \mathcal{O}_K$. (*public*)
2. Alice picks a random integer x with $0 < x < \sqrt{D}$. (*secret*)
(We know that the number of elements in the set of reduced principal ideals is usually bounded by \sqrt{D} .)
3. Bob picks a random integer y with $0 < y < \sqrt{D}$. (*secret*)
4. Alice computes \mathfrak{a} and Ψ_a such that \mathfrak{a} is the near reduced ideal of \mathfrak{g}^x with $\mathfrak{a} = (\Psi_a)\mathfrak{g}^x$. She sends \mathfrak{a} and Ψ_a to Bob. (*public*)
5. Bob computes \mathfrak{b} and Ψ_b such that \mathfrak{b} is the near reduced ideal of \mathfrak{g}^y with $\mathfrak{b} = (\Psi_b)\mathfrak{g}^y$. He sends \mathfrak{b} and Ψ_b to Alice. (*public*)

-
6. Alice computes \mathfrak{k} and Ψ_k such that \mathfrak{k} is the near reduced ideal of $((\Psi_b^{-1})\mathfrak{b})^x = \mathfrak{g}^{yx}$ with $\mathfrak{k} = (\Psi_k)\mathfrak{g}^{yx}$. (*secret*)
 7. Bob computes \mathfrak{k}' and $\Psi_{k'}$ such that \mathfrak{k}' is the near reduced ideal of $((\Psi_a^{-1})\mathfrak{a})^y = \mathfrak{g}^{xy}$ with $\mathfrak{k}' = (\Psi_{k'})\mathfrak{g}^{xy}$. (*secret*)

Since $\mathfrak{g}^{xy} = \mathfrak{g}^{yx}$ and by definition, the near reduced ideal is unique, we know $\mathfrak{k} = \mathfrak{k}'$ and $\Psi_k = \Psi_{k'}$. If the relative generators are approximated by an (f, p) representation, we cannot guarantee that $\mathfrak{k} = \mathfrak{k}'$; instead, we only know that $\mathfrak{k}' \in \{\rho^{-2}(\mathfrak{k}), \rho^{-1}(\mathfrak{k}), \mathfrak{k}, \rho(\mathfrak{k}), \rho^2(\mathfrak{k})\}$. In practice, we have found that the approximation can be made sufficiently accurate so that the keys match. To eliminate any ambiguity, a second short communication protocol is described in [JSW] that establishes a provably unique key. In the protocol, Alice computes an additional five bits based upon her key and sends this information to Bob. From this information, Bob is able to ensure that their keys match.

Note that in order for this key exchange to be secure, the set of reduced principal ideals must be sufficiently large. Discriminants D that give a high level of security are discussed in [JSW01]. An ElGamal type public-key cryptosystem and digital signatures (see [ElG85]) may also be implemented. Further details may be found in [BBT94].

Chapter 5

Survey of Improved Algorithms

Multiplication and reduction are the fundamental operations on ideals in quadratic fields and as such we want them to be as efficient as possible. Here we present several algorithms which improve on the computational efficiency of performing these operations. In this chapter, the algorithms are presented both in terms of matrices and ideals. The matrix versions of the algorithms clarify how the implementations are derived in the following chapter and both facilitate the combination of various algorithms and make these techniques immediately transferable to binary quadratic forms. Without the matrix algorithms it would be very difficult to determine how to compute (in practice) a relative generator corresponding to an ideal that has been reduced by combining two or more techniques. The ideal algorithms serve to unify the presentation in this thesis with that found throughout the literature.

5.1 Rickert's Algorithm

A computer performs its calculations using binary numbers of a fixed length called a word; for most of today's computers, the length of a word is 32 bits. Following are two example numbers converted to a computer word.

Base 10	32 bit representation
23	000000000000000000000000010111
2^{31}	100000000000000000000000000000

Numbers that fit in one word are called *single-precision*; numbers requiring more than one word are called *multi-precision*.

In [Leh38], Lehmer describes a method of speeding up the Euclidean Algorithm by working with single-precision numbers in some calculations rather than multi-precision numbers. If we are given numbers a and b (without loss of generality, $a > 0$) and want to find q and r such that $b = aq + r$, $0 \leq r < a$, we generally set $q = \lfloor b/a \rfloor$ and $r = b - aq$. Lehmer noticed that usually all that is required to compute q are the leading digits of a and b — the tails often have no effect on the result. Therefore, we should approximate a and b by taking only the leading (word size) bits of each, say \hat{a} and \hat{b} , respectively.¹ Now, set $q = \lfloor \hat{b}/\hat{a} \rfloor$ and $r = b - aq$.

This is a simplification of the original algorithm. In his paper, Lehmer generates new approximations based on the initial values for \hat{a} and \hat{b} until he knows the q values they yield are no longer correct. Then a multi-precision catch-up step is performed

¹This assumes they were the same length to start with. If not, the shorter number is padded in front with zeros.

and new approximations are taken. The full version of the modern algorithm can be found in the next chapter.

Rickert used this idea in [Ric89] to speed up the reduction algorithm for imaginary quadratic forms. Here, we let k be the maximum length of the numbers a, b and c minus the size of a computer word. Then set $\hat{a} = \lfloor a/2^k \rfloor$, $\hat{b} = \lfloor b/2^k \rfloor$, $\hat{c} = \lfloor c/2^k \rfloor$. We then reduce the form $(\hat{a}, \hat{b}, \hat{c})$ and combine the unimodular matrices used in that reduction to give a single transformation matrix which is then applied to the full form (a, b, c) . We continue this procedure until (a, b, c) is reduced. The advantage is that most of the operations and reduction calculations are performed on single or double precision numbers with comparatively few operations needing to be computed on the full multi-precision form.

We apply this technique to an ideal $\mathfrak{a}_i = (Q_i, P_i)$ by taking single precision approximations \hat{Q}_i , \hat{P}_i and \hat{R}_i of Q_i , P_i and R_i . Let $\hat{D}_i = \hat{P}_i^2 - \hat{Q}_i \hat{R}_i$, $\alpha_i = \frac{P_i - \sqrt{D}}{Q_i}$, $\hat{\alpha}_i = \frac{\hat{P}_i - \sqrt{\hat{D}_i}}{\hat{Q}_i}$ and reduce (\hat{Q}_i, \hat{P}_i) in $\mathbb{Q}(\sqrt{\hat{D}_i})$. From this procedure we obtain $\hat{\psi}_{i+1} = a_i + c_i \hat{\alpha}_i$ such that $(\hat{\psi}_{i+1})\hat{\mathfrak{a}}_i$ is reduced in $\mathbb{Q}(\sqrt{\hat{D}_i})$. Using the coefficients a_i and c_i , compute $\psi_{i+1} = a_i + c_i \bar{\alpha}_i$ and partially reduce \mathfrak{a}_i by taking $\mathfrak{a}_{i+1} = (\psi_{i+1})\mathfrak{a}_i$.

Even if $D_i > 0$, \hat{D}_i need not be positive if a naive approach to computing \hat{D}_i is used. For a simple example, consider $Q_i = 28$, $P_i = 23$, $R_i = 17$ and $D_i = 53$. If $k = 2$ then $\hat{Q}_i = \lfloor 28/2^2 \rfloor = 7$, $\hat{P}_i = \lfloor 23/2^2 \rfloor = 5$, $\hat{R}_i = \lfloor 17/2^2 \rfloor = 4$ and $\hat{D}_i = 5^2 - (7)(4) = -3$. Either the algorithm used in the single-precision reductions must take this into account or a different computation for \hat{D}_i must be used. Some possibilities are discussed in the following chapter.

Conceptually, if $\hat{D}_i > 0$ and Lagrange reduction (Algorithm 3.3) is used to reduce (\hat{Q}_i, \hat{P}_i) , this procedure embeds the partial quotients $[\tilde{q}_0, \tilde{q}_1, \dots, \tilde{q}_j]$ used in the

reduction of $(\widehat{Q}_i, \widehat{P}_i)$ into the reduction of (Q_i, P_i) so that

$$\frac{P_i + \sqrt{D}}{Q_i} = \left[\tilde{q}_0, \tilde{q}_1, \dots, \tilde{q}_j, \frac{P_{i+1} + \sqrt{D}}{Q_{i+1}} \right] .$$

When we compute the approximations \widehat{Q}_i , \widehat{P}_i and \widehat{R}_i , if one of them is 0, or if $(\widehat{Q}_i, \widehat{P}_i)$ is a reduced ideal, a full multi-precision reduction step is taken on (Q_i, P_i) instead, and then the procedure is continued with approximations. We continue this process until a fully reduced ideal \mathfrak{a}_r is obtained. The procedure is given in detail as Algorithms 5.1 and 5.2.

Algorithm 5.1 Rickert–Style Reduction Algorithm (Matrix)

Input: $\mathfrak{A}_0 = \begin{pmatrix} Q_0 & P_0 \\ P_0 & R_0 \end{pmatrix}$, D **Output:** \mathfrak{A}_i , N such that $\mathfrak{A}_i = \phi_N(\mathfrak{A}_0)$ and (Q_i, P_i) is reduced

[Initialization]

 $i \leftarrow 0$

[Reduction]

while (Q_i, P_i) is not reduced **do** $k \leftarrow \max(\max(\lceil \log_2 Q_i \rceil, \lceil \log_2 P_i \rceil, \lceil \log_2 R_i \rceil) - (\text{word size}), 0)$ $\widehat{\mathfrak{A}}_i \leftarrow \begin{pmatrix} \lfloor Q_i/2^k \rfloor & \lfloor P_i/2^k \rfloor \\ \lfloor P_i/2^k \rfloor & \lfloor R_i/2^k \rfloor \end{pmatrix}$ $\widehat{D}_i \leftarrow \widehat{P}_i^2 - \widehat{Q}_i \widehat{R}_i$ **if** an entry of $\widehat{\mathfrak{A}}_i$ is 0 **or** $(\widehat{Q}_i, \widehat{P}_i)$ is reduced in $\mathbb{Q}(\sqrt{\widehat{D}_i})$ **then**

[Full multi-precision reduction]

 $q_i \leftarrow \left\lfloor \frac{P_i + \sqrt{D}}{Q_i} \right\rfloor$ $N_i \leftarrow \begin{pmatrix} q_i & -1 \\ -1 & 0 \end{pmatrix}$ $\mathfrak{A}_{i+1} \leftarrow (-1)N_i^T \mathfrak{A}_i N_i$ **else**

[Single-precision reduction]

 $\widehat{\mathfrak{A}}_{i+1}, N_i \leftarrow \text{Algorithm 3.2 or 3.6}(\widehat{\mathfrak{A}}_i, \widehat{D}_i)$ $\mathfrak{A}_{i+1} \leftarrow (\det N_i)N_i^T \mathfrak{A}_i N_i$ $i \leftarrow i + 1$ **return** \mathfrak{A}_i , $N = \prod_{j=0}^{i-1} N_j$

Algorithm 5.2 Rickert-Style Reduction Algorithm (Ideal)

Input: $\mathfrak{a}_0 = (Q_0, P_0)$, D **Output:** \mathfrak{a}_i , Ψ_i such that $\mathfrak{a}_i = (\Psi_i)\mathfrak{a}_0$ and \mathfrak{a}_i is reduced

[Initialization]

 $i \leftarrow 0$ $\psi_0 \leftarrow 1$ $R_0 \leftarrow \frac{P_0^2 - \sqrt{D}}{Q_0}$

[Reduction]

while \mathfrak{a}_i is not reduced **do** $k \leftarrow \max(\max(\lceil \log_2 Q_i \rceil, \lceil \log_2 P_i \rceil, \lceil \log_2 R_i \rceil) - (\text{word size}), 0)$ $\widehat{Q}_i \leftarrow \lfloor Q_i/2^k \rfloor$, $\widehat{P}_i \leftarrow \lfloor P_i/2^k \rfloor$, $\widehat{R}_i \leftarrow \lfloor R_i/2^k \rfloor$ $\widehat{D}_i \leftarrow \widehat{P}_i^2 - \widehat{Q}_i \widehat{R}_i$ **if** one of \widehat{Q}_i , \widehat{P}_i or \widehat{R}_i is 0 **or** $(\widehat{Q}_i, \widehat{P}_i)$ is reduced in $\mathbb{Q}(\sqrt{\widehat{D}_i})$ **then**

[Full multi-precision reduction]

 $q_i \leftarrow \left\lfloor \frac{P_i + \sqrt{D}}{Q_i} \right\rfloor$ $\psi_{i+1} \leftarrow \overline{\alpha}_i - q_i$ **else**

[Single-precision reduction]

 $\psi_{i+1} \leftarrow$ Relative generator (with $\widehat{\alpha}_i$ replaced by $\overline{\alpha}_i$) from reducing $\widehat{\alpha}_i$ in $\mathbb{Q}(\sqrt{\widehat{D}_i})$ using Algorithm 3.2 or 3.6. $\mathfrak{a}_{i+1} \leftarrow (\psi_{i+1})\mathfrak{a}_i$ $i \leftarrow i + 1$ **return** \mathfrak{a}_i , $\Psi_i = \prod_{j=0}^i \psi_j$

5.2 Schönhage's Algorithm

The divide and conquer technique was used by Schönhage [Sch91] to give an asymptotically fast method of reducing forms. We present the algorithm here in terms of ideals and show how to compute the relative generator.

To use divide and conquer, the problem of reducing an ideal must be somehow split into smaller subproblems. The idea Schönhage had was to find an ideal that is *minimal above* some value s .

Definition 5.1. (Q, P) is *minimal above* $s > 0$ if

1. $Q, P, R \geq s$ and
2. (a) $Q - 2P + R < s$ or
(b) $P - Q < s$ and $P - R < s$.

We will show that if an ideal is minimal above 1 then it is at most one step from being reduced. Before looking into the details of this method, we present an overview of the algorithm. To satisfy the first condition of Definition 5.1 for any value of s , we need Q, P and R positive to start with; this is accomplished with Algorithm 5.4. We then find an ideal minimal above 1. Lastly, we do the final reduction step and terminate. The main components are combined as Algorithm 5.3. The algorithm returns a reduced ideal and relative generator.

Make Q, P and R Positive

The first step in making Q, P and R positive is to ensure $Q + R > 0$. If $Q + R < 0$ then negate Q, P and R . We obtain a rationale for this approach by considering the

Algorithm 5.3 Schönage Reduction (Matrix)**Input:** $\mathfrak{A} = \begin{pmatrix} Q & P \\ P & R \end{pmatrix}, D$ **Output:** \mathfrak{A}', N such that $\mathfrak{A}' = \phi_N(\mathfrak{A})$ and (Q', P') is reduced

$\mathfrak{A}', N_1 \text{ sign} \leftarrow \text{Algorithm 5.4 } (\mathfrak{A}) \text{ \{Make positive\}}$
 $\mathfrak{A}', N_2 \leftarrow \text{Algorithm 5.5 } (\mathfrak{A}', -1) \text{ \{Main reduction\}}$
 $\mathfrak{A}', N_3 \leftarrow \text{Algorithm 3.2 } (\mathfrak{A}', D) \text{ \{Final Lagrange reduction steps\}}$
 $N \leftarrow N_1 N_2 N_3$
 $\mathfrak{A}' \leftarrow \text{sign} \cdot \mathfrak{A}'$
return \mathfrak{A}', N

associated matrix $\begin{pmatrix} Q & P \\ P & R \end{pmatrix}$. If $\begin{pmatrix} Q' & P' \\ P' & R' \end{pmatrix} = \phi_N \begin{pmatrix} Q & P \\ P & R \end{pmatrix}$ then $\begin{pmatrix} -Q' & -P' \\ -P' & -R' \end{pmatrix} = \phi_N \begin{pmatrix} -Q & -P \\ -P & -R \end{pmatrix}$, that is, the steps are the same to reduce (Q, P) as to reduce $(-Q, -P)$.

Second, we want to ensure that $P > 0$. If $P < 0$, we use the matrix $N = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ and corresponding function ϕ_N from Definition 3.4 to see that $(Q, P) \sim (R, -P)$. From Equation (3.4), this transformation is realized by $(R, -P) = \left(\frac{\sqrt{D}-P}{Q}\right)(Q, P)$.

At this point we know that $Q + R > 0$ which guarantees that at most one of Q or R is negative. If $Q < 0$, then using ϕ_N where $N = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ we set $Q \leftarrow Q + 2P + R$, $P \leftarrow P + R$, and leave R unchanged. Since $Q + R, P$, and R were positive, we know the new values for Q, P , and R are positive. A similar argument works with the matrix $N = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ if $R < 0$. The full algorithm to make the ideal parameters positive is given as Algorithm 5.4.

Monotone Reduction

To find an ideal minimal above 1 we use the divide and conquer MR algorithm (Algorithm 5.5). The first step is to partially reduce (Q, P) until it is minimal above

Algorithm 5.4 Make Positive (Matrix)

Input: $\mathfrak{A} = \begin{pmatrix} Q & P \\ P & R \end{pmatrix}$ **Output:** \mathfrak{A}' , N , sign, such that all entries of \mathfrak{A}' are positive and $\mathfrak{A}' = (\text{sign})\phi_N(\mathfrak{A})$

[Initialization]

 $\mathfrak{A}' \leftarrow \mathfrak{A}$, $N \leftarrow I_2$ [Ensure $Q' + R' \geq 0$ and $P' \geq 0$]**if** $(Q' + R' < 0)$ **then** $\mathfrak{A}' \leftarrow -\mathfrak{A}'$ sign $\leftarrow -1$ **else**sign $\leftarrow 1$ **if** $(P' < 0)$ **then** $\mathfrak{A}' \leftarrow \begin{pmatrix} R' & -P' \\ -P' & Q' \end{pmatrix}$ $N \leftarrow N \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ [Ensure Q' , P' , R' are all positive]{ $Q' + R' \geq 0$ implies at most one of Q' or R' is negative. }**if** $(Q' < 0)$ **then** $\mathfrak{A}' \leftarrow \begin{pmatrix} Q' + 2P' + R' & P' + R' \\ P' + R' & R' \end{pmatrix}$ $N \leftarrow N \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ **else if** $(R' < 0)$ **then** $\mathfrak{A}' \leftarrow \begin{pmatrix} Q' & P' + Q' \\ P' + Q' & Q' + 2P' + R' \end{pmatrix}$ $N \leftarrow N \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ **return** \mathfrak{A}' , N , sign

$(1 + \max(Q, 2P, R))/2$. To achieve this partial reduction we recurse into MR with larger s values until at some point only a few reduction steps are needed to reduce the ideal minimal above that s value. Once this recursion unwinds, the algorithm is

called again and asked to reduce, minimal above 1, the ideal that was just partially reduced.

Giving a bit more detail, suppose the MR algorithm is called with the parameters Q, P, R, s . If Q, P, R are all less than $4s$ then we do a few simple steps, which we will explain later, until (Q, P) is minimal above s ; otherwise, we recurse by increasing the threshold by $\max(Q, 2P, R)/2$. That is, we choose a new value $s' = (s + \max(Q, 2P, R))/2$ and execute $Q', P', R', \psi_1 \leftarrow \text{MR}(Q, P, R, s')$. This gives (Q', P') minimal above s' . Now, a few reduction steps are taken (producing relative generator ψ_2) until Q', P' and R' are all less than s' . Then MR is called again with the original s but the new parameters corresponding to the partially reduced ideal. Thus, we execute $Q', P', R', \psi_3 \leftarrow \text{MR}(Q', P', R', s)$. We compute the relative generator $\Psi \leftarrow \psi_1 \psi_2 \psi_3$ and return the values Q', P', R', Ψ .

In the actual algorithm we often work with only the leading bits of Q, P and R to speed up calculations in a style similar to Lehmer's GCD algorithm. The full algorithm for monotone reduction used in the recursion is annotated and given as Algorithm 5.5. In the following section we explain how steps 5 and 9 use Algorithm 5.6 to reduce the ideal corresponding to \mathfrak{A}' ; these are the only steps in which the ideal is being reduced.

Algorithm 5.5 Monotone Reduction — MR (Matrix)

Input: $\mathfrak{A} = \begin{pmatrix} Q & P \\ P & R \end{pmatrix}, m$

Output: \mathfrak{A}', N such that $\mathfrak{A}' = \phi_N(\mathfrak{A})$ and (Q', P') is minimal above 2^m

[Initialization]

$\mathfrak{A}' \leftarrow \mathfrak{A}, N \leftarrow I_2$

[1. If sufficiently small in size, skip to end]

if $\min(Q', 2P', R') \geq 2^{m+2}$ then

```

[2. Calculate size of operands and split if necessary ]
 $n \leftarrow \max(\lfloor \log_2 Q' \rfloor, \lfloor \log_2 2P' \rfloor, \lfloor \log_2 R' \rfloor) - m + 1$ 
if  $m \leq n$  then
     $m' \leftarrow m, p \leftarrow 0$ 
else
     $m' \leftarrow n, p \leftarrow m - n + 1$ 
    Split  $\mathfrak{A}'$  as  $\mathfrak{A}' = 2^p \mathfrak{A}_1 + \mathfrak{A}_0$     ( $0 \leq Q_0, P_0, R_0 < 2^p$ )
     $\mathfrak{A}' \leftarrow \mathfrak{A}_1$ 

[3. Calculate half-way point ]
 $h \leftarrow m' + \lfloor n/2 \rfloor$ 
if  $\min(Q', 2P', R') \geq 2^h$  then
    [4. Recursive call — reduce  $(Q, P)$  minimal above  $2^h$  ]
     $\mathfrak{A}', N \leftarrow \text{MR}(\mathfrak{A}', h)$ 

[ 5. Reduce until  $\max(Q', 2P', R') < 2^h$  ]
while  $\max(Q', 2P', R') \geq 2^h$  do
    if  $(Q', P')$  is minimal above  $2^{m'}$  then
        break
    else
         $\mathfrak{A}', N' \leftarrow \text{Algorithm 5.6 } (\mathfrak{A}', m')$ 
         $N \leftarrow NN'$ 
if  $(Q', P')$  is not minimal above  $2^{m'}$  then
    [ 6-7. Recursive call — reduce  $(Q', P')$  minimal above  $m'$  ]
     $\mathfrak{A}', N' \leftarrow \text{MR}(\mathfrak{A}', m')$ 
     $N \leftarrow NN'$ 

    [ 8. Update tails ]
    if  $p > 0$  then
         $\mathfrak{A}' \leftarrow 2^p \mathfrak{A}' + N^T \mathfrak{A}_0 N$ 
    {from step 5}
    {from step 1}

[ 9. Reduce above  $m$  ]
while  $(Q', P')$  is not minimal above  $2^m$  do
     $\mathfrak{A}', N' \leftarrow \text{Algorithm 5.6 } (\mathfrak{A}', m')$ 
     $N \leftarrow NN'$ 

return  $\mathfrak{A}', N$ 

```

Simple Steps

Reduction in the MR algorithm is accomplished via the use of two basic steps. These reduction steps are designed so that the ideal parameters Q , P , and R remain positive. The first of these is called a *low step*. Here, Q remains unchanged and the reduction is

$$Q' \leftarrow Q , \tag{5.1}$$

$$P' \leftarrow P - Q ,$$

$$R' \leftarrow Q - 2P + R .$$

The second is called a *high step* and it is symmetric, leaving R unchanged with the reduction

$$Q' \leftarrow Q - 2P + R ,$$

$$P' \leftarrow P - R ,$$

$$R' \leftarrow R .$$

The matrices which effect these reductions are,

$$L = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \text{ low step} \quad \text{and} \quad H = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \text{ high step} .$$

The restriction that all of the ideal parameters remain positive provides a unique reduction path using the above steps. Observe that once it is not possible to take a

low step or high step without one of Q , P or R becoming negative, (Q, P) is minimal above 1 and hence almost reduced. To prove these claims, observe that at each stage, an entry in the subsequent reduction is $Q - 2P + R$. If this is positive then

$$\begin{aligned} (Q - P) + (R - P) &> 0 \\ \Rightarrow (P - Q) + (P - R) &< 0 . \end{aligned}$$

This implies that at most one of $P - Q$ and $P - R$ can be positive. Choosing the positive course, we have a unique reduction path. The Divide and Conquer technique requires a generalization of this idea. Instead of simply requiring that Q , P , and R be positive, we require that they be greater than 2^m for some $m \in \mathbb{Z}$. As we will soon show, this criterion also provides a unique reduction path and we observe from Definition 5.1 that once it is not possible to take a low step or high step without one of Q , P or R becoming less than 2^m , (Q, P) is minimal above 2^m . Given $Q, P, R > 2^m$,

$$\begin{aligned} Q - 2P + R &> 2^m \\ \Rightarrow (P - Q) + (P - R) &< -2^m \\ \Rightarrow \text{At most one of } P - Q \text{ and } P - R &\text{ is greater than } 2^m . \end{aligned}$$

Since a low step requires the calculation of $P - Q$ and a high step requires $P - R$, only one of these steps satisfies the condition that all parameters be greater than 2^m .

We may make these steps more efficient by combining consecutive low steps as $\begin{pmatrix} 1 & -q \\ 0 & 1 \end{pmatrix}$. This gives the formulas

$$Q' \leftarrow Q ,$$

$$P' \leftarrow P - qQ ,$$

$$R' \leftarrow q^2Q - 2qP + R .$$

We may find the optimal value of q so that all coefficients are greater than 2^m by seeing that $D = P'^2 - QR' \Rightarrow P'^2 \geq D + 2^mQ$, if $R' \geq 2^m$. Now we have two conditions on P' :

$$1) P' = \underbrace{P - qQ}_{P'^2 \geq D + 2^mQ} \geq \sqrt{D + 2^mQ} \Rightarrow q \leq \frac{P - \sqrt{D + 2^mQ}}{Q}$$

$$2) P' = \underbrace{P - qQ}_{\text{minimal above constraint}} \geq 2^m \Rightarrow q \leq \frac{P - 2^m}{Q}$$

Hence, the optimal value of q is

$$q = \left\lfloor \frac{P - t}{Q} \right\rfloor , \quad t = \max(\sqrt{D + 2^mQ}, 2^m) .$$

Consecutive high steps may be combined with q found symmetrically as

$$q = \left\lfloor \frac{P - t}{R} \right\rfloor , \quad t = \max(\sqrt{D + 2^mR}, 2^m) .$$

Given Q, P and R , one combined set of low steps or one combined set of high steps is referred to as a simple step. Algorithms 5.6 and 5.7 perform the simple step operation. In the following chapter, the implementation is based on Algorithm 5.6 while Algorithm 5.7 is provided to clarify how the ideal and relative generator are affected by each simple step.

Algorithm 5.6 Simple Step Above 2^m (Matrix)

Input: $\mathfrak{A}_i = \begin{pmatrix} Q_i & P_i \\ P_i & R_i \end{pmatrix}$, m where (Q_i, P_i) is an ideal that is to be reduced one simple step above 2^m

Output: \mathfrak{A}_{i+1} , N such that $\mathfrak{A}_{i+1} = \phi_N(\mathfrak{A}_i)$ and (Q_{i+1}, P_{i+1}) has been reduced a simple step

$D \leftarrow P_i^2 - Q_i R_i$ {Store in a lookup table indexed by Q_i, P_i, R_i }

if $(P_i > Q_i)$ **then**

 [Low step]

$t \leftarrow \max(\lceil \sqrt{D + 2^m Q_i} \rceil, 2^m)$

$q \leftarrow \left\lfloor \frac{P_i - t}{Q_i} \right\rfloor$

$N \leftarrow \begin{pmatrix} 1 & -q \\ 0 & 1 \end{pmatrix}$

else

 [High step]

$t \leftarrow \max(\lceil \sqrt{D + 2^m R_i} \rceil, 2^m)$

$q \leftarrow \left\lfloor \frac{P_i - t}{R_i} \right\rfloor$

$N \leftarrow \begin{pmatrix} 1 & 0 \\ -q & 1 \end{pmatrix}$

$\mathfrak{A}_{i+1} \leftarrow N^T \mathfrak{A}_i N$

return \mathfrak{A}_{i+1}, N

Finishing The Reduction

With these definitions in place, we are now able to show that if the MR algorithm is called with a value between 0 and 1 it yields an ideal within one step of being reduced.

Proposition 5.2. *An ideal (Q, P) minimal above 1 is within one step of being reduced.*

Algorithm 5.7 Simple Step Above 2^m (Ideal)**Input:** $\mathfrak{a}_i = (Q_i, P_i), m$ **Output:** $\mathfrak{a}_{i+1} = (Q_{i+1}, P_{i+1}), \psi_{i+1}$ such that $\mathfrak{a}_{i+1} = (\psi_{i+1})\mathfrak{a}_i$ and \mathfrak{a}_{i+1} has been reduced a simple step above 2^m $D \leftarrow P_i^2 - Q_i R_i$ {Store in a lookup table indexed by Q_i, P_i, R_i }**if** $(P_i > Q_i)$ **then**

[Low step]

 $t \leftarrow \max(\lceil \sqrt{D + 2^m Q_i} \rceil, 2^m)$ $q \leftarrow \left\lfloor \frac{P_i - t}{Q_i} \right\rfloor$ $\psi_{i+1} \leftarrow 1$ **else**

[High step]

 $t \leftarrow \max(\lceil \sqrt{D + 2^m R_i} \rceil, 2^m)$ $q \leftarrow \left\lfloor \frac{P_i - t}{R_i} \right\rfloor$ $\psi_{i+1} \leftarrow 1 - q\bar{\alpha}_i$ $\mathfrak{a}_{i+1} \leftarrow (\psi_{i+1})\mathfrak{a}_i$ **return** $\mathfrak{a}_{i+1}, \psi_{i+1}$ *Proof.* Without loss of generality, assume $Q < R$.

$$D > 0 \Rightarrow P^2 > QR \Rightarrow P > Q \quad (D \text{ is not a square}) .$$

Also, this implies that condition 2b of Definition 5.1 is impossible.

 (Q, P) minimal above 1

$$\Rightarrow Q - 2P + R < 1 \quad (\text{definition})$$

$$\Rightarrow Q - 2P + R < 0 \quad (D \text{ is not square})$$

$$\Rightarrow R < 2P - Q$$

$$\begin{aligned} \Rightarrow (P - Q)^2 &= P^2 - Q(2P - Q) < P^2 - QR = D \\ \Rightarrow 0 < P - Q &< \sqrt{D} & (0 < Q < P) . \end{aligned}$$

Taking a low step (recall Equation (5.1)) yields

$$\begin{aligned} \Rightarrow 0 < P' &< \sqrt{D} & (P' = P - Q) \\ \Rightarrow 0 < |D - P'^2| &= |Q'R'| < D & (D = P'^2 - Q'R') \\ \Rightarrow Q' < \sqrt{D} &\text{ or } |R'| < \sqrt{D} \\ \Rightarrow (Q', P') &\text{ or } (R', -P') \text{ is reduced} \\ &(\text{see, for example, [WW87, Thm 4.3]}) . \end{aligned}$$

□

5.3 Schnorr and Seysen's Algorithm

In November 1982, Seysen wrote an unpublished manuscript entitled “An Improved Composition Algorithm.” An updated paper under the same title, coauthored with Schnorr, was produced in August 1983. Since the paper is not generally available, a reproduction has been included as Appendix A. Although never published, this paper contains the essential idea of the later algorithms described in Sections 5.4 and 5.6.

The original algorithm was cast in the language of forms but we present it here in terms of ideals with minor corrections. Also, the notation and arguments of the proof have been changed to the standard notation of continued fractions. Note that the style of the proof has not changed — the authors were indeed using techniques equivalent to continued fraction theory although their notation would suggest that perhaps they were unaware of it.

Recall from Section 3.1 that multiplication of two reduced ideals (Q_a, P_a) and (Q_b, P_b) of the same discriminant yielding $(S)(Q_0, P_0) = (Q_a, P_a)(Q_b, P_b)$ requires the calculation of

$$\sigma S = \gcd(Q_a, Q_b, P_a + P_b) = XQ_a + YQ_b + Z(P_a + P_b) , \quad (5.2)$$

$$U = Y(P_b - P_a) + ZR_b \pmod{Q_a/S} ,$$

$$Q_0 = \frac{Q_a Q_b}{\sigma S^2} ,$$

$$P_0 = \frac{Q_b U}{\sigma S} + P_b .$$

By Equation (3.3) and [WW87, Thm 4.3], reducing (Q_0, P_0) involves finding e and g such that $|Q_r| = |g^2 Q_0 + 2egP_0 + g^2 R_0| < \sqrt{D}$.

Using Equations (5.2) we may derive

$$\begin{aligned} Q_r &= e^2 Q_0 + 2egP_0 + g^2 R_0 \\ &= \frac{e^2 Q_a Q_b}{S^2} + 2 \frac{eg Q_b U}{S} + \frac{g^2 Q_b U^2}{Q_a} + 2egP_b + 2 \frac{g^2 S P_b U}{Q_a} + \frac{g^2 S^2 P_b^2}{Q_a Q_b} - \frac{g^2 S^2 D}{Q_a Q_b} \\ &= \frac{S}{Q_a} \left(\frac{Q_b}{S} \left(\frac{e^2 Q_a^2}{S^2} + 2 \frac{eg Q_a U}{S} + g^2 U^2 \right) + 2gP_b \left(\frac{e Q_a}{S} + gU \right) + g^2 S \frac{P_b^2 - D}{Q_b} \right) \\ &= \frac{S}{Q_a} \left(\frac{Q_b}{S} \left(e \frac{Q_a}{S} + gU \right)^2 + 2gP_b \left(e \frac{Q_a}{S} + gU \right) + g^2 S R_b \right) . \end{aligned} \quad (5.3)$$

The authors recognized that $\left(e \frac{Q_a}{S} + gU \right)$ is the dominant term in this equation and so decreasing its magnitude would decrease the size of Q_r . The extended Euclidean algorithm on input of Q_a/S and U outputs numbers e and g such that $\gcd(Q_a/S, U) = e(Q_a/S) + gU$. This is exactly the dominant term in Equation (5.3) and the GCD is the smallest possible integer value (in absolute value) it can take. De-

veloping the simple continued fraction of $U/(Q_a/S)$ is essentially the same procedure and also computes the numbers e and g .

To decrease the size of Q_r , we show that we do not need to develop the entire continued fraction expansion but only a portion of it. Let $[q_0, q_1, \dots, q_m]$ be the partial quotients of the simple continued fraction expansion of the rational number $\frac{U}{Q_a/S}$, define A_i and B_i as in Equation (3.10), and

$$C_{-2} = U, \quad C_{-1} = Q_a/S, \quad C_i = C_{i-2} - q_i C_{i-1} . \quad (5.4)$$

By Equation (3.12), $A_i B_{i-1} - A_{i-1} B_i = (-1)^{i-1}$; therefore, by Proposition 3.2 (with $a \leftarrow B_{i-1}$, $b \leftarrow -A_{i-1}$, $c \leftarrow -B_i$, $d \leftarrow A_i$), if we let $\varepsilon = (-1)^{i-1}$, then (Q_0, P_0) is equivalent to

$$(\varepsilon(A_i^2 Q_0 - 2A_i B_i P_0 + B_i^2 R_0), \varepsilon(-A_{i-1} A_i Q_0 + (A_i B_{i-1} + A_{i-1} B_i) P_0 - B_{i-1} B_i R_0)) . \quad (5.5)$$

If we find r such that

$$C_r \leq \sqrt{\frac{Q_a \sqrt{D}}{Q_b}} < C_{r-1} , \quad (5.6)$$

and combine this with the fact that the remainders $\{C_r\}$ are strictly decreasing, we obtain the following relations

$$\begin{aligned} B_r C_{r-1} + B_{r-1} C_r &= C_{-1} \quad (\text{continued fraction theory}) \\ \Rightarrow B_r C_r &< B_r C_{r-1} \leq B_r C_{r-1} + B_{r-1} C_r = \frac{Q_a}{S} \end{aligned} \quad (5.7)$$

$$\begin{aligned}
\Rightarrow B_r &\leq \frac{Q_a/S}{C_{r-1}} < \frac{Q_a/S}{\sqrt{\frac{Q_a\sqrt{D}}{Q_b}}} \quad (\text{by (5.6)}) \\
\Rightarrow B_r^2 &\leq \frac{Q_a Q_b}{S^2 \sqrt{D}} \tag{5.8}
\end{aligned}$$

which will soon be used to place a bound on $|Q_r|$ but we first require the following lemma from continued fraction theory.

Lemma 5.3. *Let C_{-2} and C_{-1} be given and A_i , B_i and C_i defined as previously stated, then*

$$C_i = (-1)^{i-1}(A_i C_{-1} - B_i C_{-2}) .$$

Proof (by strong induction).

Base cases:

$$\begin{aligned}
i = -2 & \quad (-1)^{-3}(A_{-2}C_{-1} - B_{-2}C_{-2}) = -(-C_{-2}) = C_{-2} \\
i = -1 & \quad (-1)^{-2}(A_{-1}C_{-1} - B_{-1}C_{-2}) = C_{-1}
\end{aligned}$$

Induction hypothesis: Let $k \geq 0 \in \mathbb{Z}$ be fixed and assume

$$C_i = (-1)^{i-1}(A_i C_{-1} - B_i C_{-2}) \quad (-2 \leq i < k) .$$

Under this assumption, we get

$$C_k = C_{k-2} - q_k C_{k-1}$$

$$\begin{aligned}
&= (-1)^{k-3}(A_{k-2}C_{-1} - B_{k-2}C_{-2}) - q_k(-1)^{k-2}(A_{k-1}C_{-1} - B_{k-1}C_{-2}) \\
&= (-1)^{k-1}((A_{k-2}C_{-1} - B_{k-2}C_{-2}) + (q_k A_{k-1}C_{-1} - q_k B_{k-1}C_{-2})) \\
&= (-1)^{k-1}((q_k A_{k-1} + A_{k-2})C_{-1} - (q_k B_{k-1} + B_{k-2})C_{-2}) \\
&= (-1)^{k-1}(A_k C_{-1} - B_k C_{-2}) . \quad \square
\end{aligned}$$

Applying this lemma we obtain

$$\begin{aligned}
|C_r| &= |A_r C_{-1} - B_r C_{-2}| \\
&= \left| A_r \frac{Q_a}{S} - B_r U \right| .
\end{aligned}$$

Substituting these values into Equation (5.3) we have

$$\begin{aligned}
|Q_r| &\leq \left| \frac{S}{Q_a} \left(\frac{Q_b}{S} C_r^2 + 2P_b B_r C_r + B_r^2 S R_b \right) \right| \\
&\leq \left| \frac{S}{Q_a} \left(\frac{Q_b}{S} \frac{Q_a \sqrt{D}}{Q_b} + 2P_b \frac{Q_a}{S} + \frac{Q_a Q_b}{S^2 \sqrt{D}} S R_b \right) \right| \quad (\text{by (5.6 - 5.8)}) \\
&\leq \left| \sqrt{D} + 2P_b + \frac{Q_b R_b}{\sqrt{D}} \right| \\
&< 4\sqrt{D} \quad (\text{since } (Q_b, P_b) \text{ is reduced}) .
\end{aligned}$$

By [WW87, Cor. 4.2.1], (Q_r, P_r) is within a few steps of being reduced. In their implementation, Schnorr and Seysen compute (Q_0, P_0) and then reduce it using Equation (5.5). The saving of this method over reducing (Q_0, P_0) classically is that the operations are simpler since we are taking the continued fraction of a rational number instead of a quadratic irrational. Also, the calculation of the values Q_i , P_i , and R_i at each stage is traded for the calculation of A_i and B_i and so we are working with

smaller numbers. We will see later that this is essentially the idea that Atkin developed in his improvement to Shanks' algorithm except that he also found a way to compute (Q_r, P_r) directly (without first needing to compute (Q_0, P_0)) with intermediate operands not usually exceeding \sqrt{D} . Thus, this algorithm will not be implemented in the following chapter.

5.4 A New Algorithm For Ideal Reduction

Recall from Section 3.3.2 that the Lagrange algorithm to reduce an ideal (Q_0, P_0) of positive discriminant is achieved through the continued fraction expansion of $(P_0 + \sqrt{D})/Q_0$. The sequence of ideals (Q_i, P_i) is generated by

$$\frac{P_0 + \sqrt{D}}{Q_0} = \left[q_0, q_1, \dots, q_i, \frac{P_{i+1} + \sqrt{D}}{Q_{i+1}} \right],$$

where the values q_i, Q_{i+1}, P_{i+1} are obtained from Algorithm 3.1.

Observe that after multiplication of two reduced ideals, the values of P_0 and Q_0 in the resulting ideal (Q_0, P_0) typically have magnitude D ; hence, adding \sqrt{D} to P_0 does not usually affect the value of the quotient obtained. That is, $P_0/Q_0 \approx (P_0 + \sqrt{D})/Q_0$. This means we may eliminate \sqrt{D} from the computations for some time. The question we ask then is, "At what point do we need to include \sqrt{D} again?" Surprisingly, we do not need to include it at all.

Suppose the partial quotients of the continued fraction expansion of P_0/Q_0 are given by $[\tilde{q}_0, \tilde{q}_1, \dots, \tilde{q}_m]$. We embed a portion of this sequence into the continued

fraction expansion of $(P_0 + \sqrt{D})/Q_0$:

$$\frac{P_0 + \sqrt{D}}{Q_0} = \left[\tilde{q}_0, \tilde{q}_1, \dots, \tilde{q}_r, \frac{P_{r+1} + \sqrt{D}}{Q_{r+1}} \right] \quad (r \leq m)$$

where the Q_{r+1} and P_{r+1} values could be computed at each step along the way, or by formulae presented later starting in Equation (5.9). Note that this was also the case in the Schnorr-Seysen algorithm except that we embedded the partial quotients of $\frac{U}{Q_a/S}$ instead.

Although the quotients may differ from the “correct” expansion, each new ideal is of course still equivalent to the previous one. What we will show is that this process reduces the size of Q_i such that $|Q_{r+1}| < \sqrt{D}$ for some r , and further that this guarantees (Q_{r+1}, P_{r+1}) is reduced. In addition, we show that the technique used to reduce ideals of positive discriminant also reduces ideals of negative discriminant.

In the following discussion, we assume that Q_0 is positive. This poses no problem since $(Q_0, P_0) = (-Q_0, P_0)$.

Theorem 5.4. *Given an ideal (Q_0, P_0) of positive discriminant, let $[\tilde{q}_0, \tilde{q}_1, \dots, \tilde{q}_m]$ be the partial quotients of the simple continued fraction expansion of P_0/Q_0 . Embedding these quotients into the continued fraction expansion*

$$\frac{P_0 + \sqrt{D}}{Q_0} = \left[\tilde{q}_0, \tilde{q}_1, \dots, \tilde{q}_r, \frac{P_{r+1} + \sqrt{D}}{Q_{r+1}} \right]$$

yields a reduced ideal (Q_{r+1}, P_{r+1}) for some $r \leq m$.

Proof. This is similar to Lagrange reduction except that the quotients \tilde{q}_i as defined

above are used instead. Let

$$\begin{aligned} A_{-2} &= 0, & A_{-1} &= 1, & A_i &= \tilde{q}_i A_{i-1} + A_{i-2} , \\ B_{-2} &= 1, & B_{-1} &= 0, & B_i &= \tilde{q}_i B_{i-1} + B_{i-2} . \end{aligned}$$

Then by Equations (3.7), (3.9) and (3.11) we know

$$\begin{aligned} G_i &= Q_0 A_i - P_0 B_i , \\ Q_{i+1} &= (-1)^{i+1} \left(\frac{G_i^2 - D B_i^2}{Q_0} \right) , \\ P_{i+1} &= (-1)^i \left(\frac{G_i G_{i-1} - D B_i B_{i-1}}{Q_0} \right) = \frac{G_i - Q_{i+1} B_{i-1}}{B_i} , \end{aligned} \tag{5.9}$$

and so $|Q_{r+1}| < \max \left(\frac{G_r^2}{Q_0}, \frac{D B_r^2}{Q_0} \right)$. We want to show that each of these components is less than \sqrt{D} for a suitable choice of r .

From continued fraction theory we know that A_i/B_i approximates P_0/Q_0 . In particular, we have the bound (see [Ros00, Cor. 12.3])

$$\begin{aligned} & \left| \frac{A_i}{B_i} - \frac{P_0}{Q_0} \right| < \frac{1}{B_i B_{i+1}} \\ \Rightarrow & \left| \frac{Q_0 A_i - P_0 B_i}{Q_0 B_i} \right| < \frac{1}{B_i B_{i+1}} \\ \Rightarrow & |G_i| < \frac{Q_0}{B_{i+1}} \\ \Rightarrow & G_i^2 < \frac{Q_0^2}{B_{i+1}^2} . \end{aligned} \tag{5.10}$$

Since Q_0 is positive, each $\tilde{q}_i > 0$ for $i \geq 1$ (\tilde{q}_0 might be 0 or negative) and the sequence

$\{B_i\}$ is strictly increasing for $i \geq -1$; therefore, at some point we obtain r such that

$$B_r < \frac{\sqrt{Q_0}}{\sqrt[4]{D}} < B_{r+1} . \quad (5.11)$$

This bound, along with Equation (5.10) gives

$$\frac{G_r^2}{Q_0} < \frac{Q_0}{B_{r+1}^2} < \frac{Q_0}{\left(\frac{\sqrt{Q_0}}{\sqrt[4]{D}}\right)^2} = \sqrt{D}$$

and

$$\frac{DB_r^2}{Q_0} < \frac{D\left(\frac{\sqrt{Q_0}}{\sqrt[4]{D}}\right)^2}{Q_0} = \sqrt{D} .$$

Therefore $|Q_{r+1}| < \sqrt{D}$ and so (Q_{r+1}, P_{r+1}) is reduced by [WW87, Thm 4.3]. \square

The proof and algorithm are readily adapted to the case of ideals of negative discriminant. Here, we do not have the concept of embedding the quotients into a continued fraction expansion but the proof did not depend on this intuition. We replace D by $|D|$ throughout the proof and so we have

$$\begin{aligned} Q_{r+1} &= (-1)^{r+1} \left(\frac{G_r^2 - DB_r^2}{Q_0} \right) = (-1)^{r+1} \left(\frac{G_r^2 + |D|B_r^2}{Q_0} \right) \\ \Rightarrow |Q_{r+1}| &= \left| \frac{G_r^2 + |D|B_r^2}{Q_0} \right| < \sqrt{|D|} + \sqrt{|D|} = 2\sqrt{|D|} . \end{aligned}$$

By [Coh93, Prop 5.4.3], (Q_{r+1}, P_{r+1}) is reduced within 3 steps of Algorithm 3.8.

The algorithm is presented in terms of ideals as Algorithm 5.8, and in terms of

matrices as Algorithm 5.9. We include the matrix version of the algorithm to unify the presentation of the algorithms in this thesis, facilitate the integration of this method into other reduction methods, and provide an algorithm which may easily be adapted to binary quadratic forms.

Algorithm 5.8 Efficient Ideal Reduction (Ideal)

Input: $\alpha_0 = (Q_0, P_0)$, D , where $Q_0, P_0 > 0$

Output: Ideal α_{i+1} and Ψ_{i+1} such that $\alpha_{i+1} = (\Psi_{i+1})\alpha_0$. If $D > 0$, α_{i+1} is reduced, if $D < 0$, α_{i+1} is within 3 steps of being reduced.

```

[ Initialization ]
 $i \leftarrow -2$ 
 $A_{-2} \leftarrow 0, A_{-1} \leftarrow 1, B_{-2} \leftarrow 1, B_{-1} \leftarrow 0$ 
 $C_{-2} \leftarrow P_0, C_{-1} \leftarrow Q_0$ 

[ Continued fraction expansion of  $P_0/Q_0$  ]
while  $B_{i+1} < \frac{\sqrt{Q_0}}{\sqrt[4]{|D|}}$  do
     $i \leftarrow i + 1$ 
     $q_{i+1} \leftarrow \lfloor C_{i-1}/C_i \rfloor, C_{i+1} \leftarrow C_{i-1} - q_{i+1}C_i$ 
     $A_{i+1} \leftarrow q_{i+1}A_i + A_{i-1}, B_{i+1} \leftarrow q_{i+1}B_i + B_{i-1}$ 

[ Compute  $\Psi_{i+1}$  and  $\alpha_{i+1}$  ]
 $\Psi_{i+1} \leftarrow (-1)^{i+1}(A_i - B_i\bar{\alpha}_0)$ 
 $\alpha_{i+1} \leftarrow (\Psi_{i+1})\alpha_0$ 

return  $\alpha_{i+1}, \Psi_{i+1}$ 

```

Algorithm 5.9 Efficient Ideal Reduction (Matrix)

Input: $\mathfrak{A}_0 = \begin{pmatrix} Q_0 & P_0 \\ P_0 & R_0 \end{pmatrix}, D$

Output: \mathfrak{A}_{i+1}, N such that $\mathfrak{A}_{i+1} = \phi_N(\mathfrak{A}_0)$. If $D > 0$, (Q_{i+1}, P_{i+1}) is reduced, if $D < 0$, (Q_{i+1}, P_{i+1}) is within 3 steps of being reduced.

[Initialization]

$i \leftarrow -2$

$N \leftarrow I_2$

$(C_{-2}, C_{-1}) \leftarrow (P_0, Q_0)$

[Continued fraction expansion of P_0/Q_0]

while $|N_{2,1}| < \frac{\sqrt{Q_0}}{\sqrt[4]{|D|}}$ **do**

$i \leftarrow i + 1$

$q_{i+1} \leftarrow \lfloor C_{i-1}/C_i \rfloor, C_{i+1} \leftarrow C_{i-1} - q_{i+1}C_i$

$N \leftarrow N \begin{pmatrix} q_{i+1} & -1 \\ -1 & 0 \end{pmatrix}$

[Compute \mathfrak{A}_{i+1}]

$N \leftarrow N \begin{pmatrix} q_{i+1} & -1 \\ -1 & 0 \end{pmatrix}^{-1}$ {Backup one step}

$\mathfrak{A}_{i+1} \leftarrow \phi_N(\mathfrak{A}_0)$

return \mathfrak{A}_{i+1}, N

5.5 Shanks' Algorithm – NUCOMP

In 1978, Shanks first published some thoughts on what he called the “Magic Matrix” with further details added at a conference in 1988 [Sha78, Sha89]. He published this work in terms of binary quadratic forms of negative discriminant; however, we have translated it here into the language of ideals. The matrix allows one to compute a reduced ideal equivalent to the product of two ideals without actually computing the unreduced product. Recall that the parameters of a reduced ideal are approximately the same size as \sqrt{D} and the parameters of the product of two reduced ideals are approximately the same size as D . The matrix has the innovative feature that it encapsulates all of the information about the two ideals being multiplied as well as their product, without actually requiring the computations relating to the multiplication. This allows us to compute with numbers which are only as large as \sqrt{D} and avoid computing with numbers as large as D . Shanks' motivation in developing this algorithm was that he was limited on his programmable calculator to computing with 10 significant digits. As we will see, this algorithm allowed him to work with binary quadratic forms whose discriminant was twice that size.

We will multiply two reduced ideals $\mathfrak{a}'_0 = (Q'_0, P'_0)$ and $\mathfrak{a}''_0 = (Q''_0, P''_0)$ and obtain a product that is within a few steps of being reduced. Rather than computing the unreduced product (Q_0, P_0) and then reducing it, we find ideals $\mathfrak{a}'_r = (Q'_r, P'_r)$ and $\mathfrak{a}''_r = (Q''_r, P''_r)$ equivalent to \mathfrak{a}'_0 and \mathfrak{a}''_0 (respectively) that have the property that when they are multiplied, we obtain an ideal (Q_r, P_r) that is within a few steps of being reduced. The advantage of this method is that we work with numbers of magnitude $O(\sqrt{D})$ rather than $O(D)$.

Let $\sigma S = \gcd(Q'_0, Q''_0, P'_0 + P''_0)$ and assemble the magic matrix as follows:

$$\begin{pmatrix} \frac{Q'_0}{\sigma S} & \frac{Q''_0}{\sigma S} & \frac{P'_0 + P''_0}{\sigma S} & 0 \\ U & V & W & S \end{pmatrix}.$$

We compute U, V, W by solving the following equations:

$$\begin{aligned} \frac{Q'_0}{\sigma S} V - \frac{Q''_0}{\sigma S} U &= \frac{P'_0 - P''_0}{\sigma}, \\ \frac{Q''_0}{\sigma S} W - \frac{P'_0 + P''_0}{\sigma S} V &= \frac{R'_0}{\sigma}, \\ \frac{Q'_0}{\sigma S} W - \frac{P'_0 + P''_0}{\sigma S} U &= \frac{R''_0}{\sigma}. \end{aligned} \tag{5.12}$$

In Figure 5.1 we relabel the entries of the matrix so that the follow-up is a bit simpler.

Let $S_i = P'_i + P''_i$ and $M_i = P''_i - P'_i$.

$$\begin{pmatrix} a_{i-1} & b_{i-1} & c_{i-1} & d_{i-1} \\ a_i & b_i & c_i & d_i \end{pmatrix} \begin{array}{c} \begin{array}{c} \overline{Q'_i/\sigma} \\ \overline{Q''_i/\sigma} \\ \overline{S_i/\sigma} \end{array} \\ \begin{array}{c} \overline{M_i/\sigma} \parallel \overline{R'_i/\sigma} \\ \overline{R''_i/\sigma} \end{array} \end{array}$$

Figure 5.1: Magic Matrix

What the figure indicates is that as we move from the initial ideals α'_0 and α''_0 to α'_r and α''_r , we may easily compute the parameters of the intermediate ideals. To calculate Q'_i of α'_i , compute the determinant of the 2×2 matrix formed from the first and fourth columns. To calculate Q''_i of α''_i , compute the determinant of the matrix

formed from the second and fourth columns and so on. It is not necessary, but in this manner it is possible to compute the two ideals α'_i and α''_i that are being multiplied to obtain the product $\alpha'_i \alpha''_i = (S)(Q_i, P_i)$ given by following equations:

$$\begin{aligned} \frac{Q_i}{\sigma} &= a_{i-1}b_{i-1} + c_{i-1}d_{i-1} , \\ \frac{2P_i}{\sigma} &= a_{i-1}b_i + a_i b_{i-1} + c_{i-1}d_i + c_i d_{i-1} , \\ \frac{R_i}{\sigma} &= a_i b_i + c_i d_i . \end{aligned} \tag{5.13}$$

To compute Row_{i+1} from rows Row_i and Row_{i-1} , Shanks' idea was to compute an approximation to P_i and R_i using only single precision floating point arithmetic and then with Equation (3.14) in mind, compute q_i such that $-R_i < q_i R_i - P_i \leq R_i$. He then calculated the next row of the matrix by computing

$$\text{Row}_{i+1} = \text{Row}_{i-1} - q_i \text{Row}_i .$$

The process terminates once $q_i = 0$. The product ideal is determined according to Equations (5.13) and we have an ideal that is almost reduced.

NUCOMP works with numbers approximately half the size of classical algorithms but due to the many extra computations required, its time efficiency is not significantly better. At each step, approximations are taken to P_i and R_i that require a combined six multiplications and four additions. After that, four multi-precision multiplications and subtractions in the $\text{Row}_{i+1} = \text{Row}_{i-1} - q_i \text{Row}_i$ computation are required.

Shanks' aim was not faster computations; his concern was that he be able to

work with forms of larger discriminant on his programmable calculator. Since the parameters of the forms did not exceed $\sqrt{\Delta}$ when using NUCOMP, he could compute with forms whose discriminant length was 20 digits long rather than the maximum 10 digit discriminant he would be limited to if the calculation of the unreduced product was required. The stage was set for others to build on this work and improve the time efficiency.

5.6 Improvements to NUCOMP

Upon reading Shanks' algorithm, Atkin [Atk] found some key improvements. He realized that rather than computing quotients from approximations of P_i and R_i , all that was really required was to perform a Euclidean division on the first column of the matrix until it was less than $\sqrt[4]{\Delta/4}$. This allowed him to take advantage of Lehmer's GCD algorithm, something that was not possible with Shanks' method, and also provided an earlier stopping point since Shanks continued until the q_i calculated was 0. Using the labelling from Figure 5.1, we compute q_i and a_{i+1} such that $a_{i-1} = q_i a_i + a_{i+1}$ with $0 \leq a_{i+1} < |a_i|$. We then update column 4 by setting $d_{i+1} = d_{i-1} - q_i d_i$. We don't calculate the second and third columns as we go along but once we arrive at an a_{i+1} small enough, we calculate the values $b_{i-1}, b_i, c_{i-1}, c_i$ using $a_{i-1}, a_i, d_{i-1}, d_i$. This is a huge savings since we don't need to compute an approximation to P_i and R_i each time, nor do we compute the middle two columns. Cohen improved on the computational efficiency of Atkin's formulas in [Coh93] and the algorithm was implemented and analysed for quadratic forms of negative discriminant by Düllmann in [BDW90].

To apply this algorithm to ideals of positive discriminant, it is again important to keep track of the relative generator so that a unique reduced ideal can be computed. In [vdP03], van der Poorten noticed that the distances between ideals could be worked out using Atkin's variant of the NUCOMP algorithm with very little additional effort. The technique used was similar to the method outlined in Section 3.2.2. In this case, we keep track of the standard $\{B_i\}$ convergents and use these to compute the relative generator. In fact, the $\{B_i\}$ become the fourth column in the magic matrix. This work was implemented with Jacobson in [JvdP02] in the language of quadratic forms.²

Recently, Jacobson, Scheidler and Williams have worked to bring all of these ideas together in the language of ideals [JSW].³ To simultaneously multiply and reduce two reduced ideals (Q_a, P_a) and (Q_b, P_b) , we compute S, X, Y, Z such that $\sigma S = \gcd(Q_a, Q_b, P_a + P_b) = XQ_a + YQ_b + Z(P_a + P_b)$. Then compute $U = Y(P_b - P_a) + ZR_b \pmod{Q_a/S}$. To complete the multiplication we would set $Q_0 = \frac{Q_a Q_b}{\sigma S^2}$ and $P_0 = \frac{Q_b U}{\sigma S} + P_b$; however, we don't want to do this just yet. If we actually performed this multiplication, Q_0 and P_0 would have magnitude D and then we would need to reduce them to magnitude \sqrt{D} . What we want to do instead is work with numbers whose magnitude is less than \sqrt{D} .

Instead of multiplying and then using the continued fraction expansion of $(P_0 + \sqrt{D})/Q_0$ to reduce the ideal, we find an approximation using the smaller terms that

²In this thesis we have elected to use Shanks' ordering of the magic matrix for both historical and aesthetic reasons. Readers familiar with [vdP03] may apply the following translation to use the magic matrix as presented here:

[vdP03]	a_i	b_i	c_i	d_i
This Thesis	d_i	a_i	b_i	c_i

³Readers familiar with [JSW] may use the same map as given for [vdP03]; also, the variables d_i and R have opposite sign.

we know:

$$\begin{aligned}
 & \frac{P_0 + \sqrt{D}}{Q_0} \\
 &= \frac{P_b + \frac{UQ_b}{\sigma S} + \sqrt{D}}{Q_0} \\
 &= \frac{P_b + \sqrt{D}}{Q_0} + \frac{\frac{UQ_b}{\sigma S}}{\frac{Q_a Q_b}{\sigma S^2}} \\
 &= \frac{P_b + \sqrt{D}}{Q_0} + \frac{U}{Q_a/S} .
 \end{aligned}$$

Now notice that

$$\frac{P_b + \sqrt{D}}{Q_0} \approx \frac{2\sqrt{D}}{D} = \frac{2}{\sqrt{D}} ,$$

while

$$\frac{U}{Q_a/S} \approx \frac{\sqrt{D}}{\sqrt{D}} = 1 ,$$

so $(P_0 + \sqrt{D})/Q_0$ may be approximated by $U/(Q_a/S)$ since $\frac{P_b + \sqrt{D}}{Q_0}$ has little effect on the result. Reminiscent of Sections 5.3 and 5.4, let $[\tilde{q}_0, \tilde{q}_1, \dots, \tilde{q}_m]$ be the partial quotients of the simple continued fraction expansion of $\frac{U}{Q_a/S}$. Embedding these quotients into the continued fraction expansion

$$\frac{P_0 + \sqrt{D}}{Q_0} = \left[\tilde{q}_0, \tilde{q}_1, \dots, \tilde{q}_r, \frac{P_{r+1} + \sqrt{D}}{Q_{r+1}} \right]$$

yields a sequence of equivalent ideals. Jacobson, Scheidler and Williams show that if we use initial values

$$\begin{aligned}
 a_{-1} &= \frac{Q_a}{S} & a_0 &= U \\
 b_{-1} &= \frac{Q_b}{\sigma S} & b_0 &= \frac{1}{a_{-1}} (b_{-1}U + (P_b - P_a)) \\
 c_{-1} &= P_a + P_b & c_0 &= \frac{1}{a_{-1}} (c_{-1}U + \sigma S R_b) \\
 d_{-1} &= 0 & d_0 &= 1
 \end{aligned}$$

and calculate the continued fraction expansion of a_{-1}/a_0 while also updating the d_i , then at any point we may compute

$$\begin{aligned}
 b_j &= \frac{1}{a_{-1}} (a_j b_{-1} + d_j (P_b - P_a)) \quad , \\
 c_j &= \frac{1}{a_{-1}} (a_j c_{-1} + \sigma d_j S R_b) \quad .
 \end{aligned}$$

Putting this in the context of Shanks' Magic Matrix, we have the following initial entries:

$$\begin{pmatrix} Q_a/S & Q_b/\sigma S & P_a + P_b & 0 \\ U & b_0 & c_0 & 1 \end{pmatrix}$$

We may compute the product (Q_i, P_i) with the following formulas (note the improved formula for P_{i+1} — compare with (5.13)):

$$\begin{aligned}
 Q_{i+1} &= (-1)^{i+1} (a_i b_i + c_i d_i) \\
 P_{i+1} &= (-1)^{i+1} (a_{i-1} b_i + c_{i-1} d_i) + P_b
 \end{aligned}$$

Due to the amalgamation of multiplication and reduction, presenting a theoretical version of the algorithm in terms of matrices or ideals would serve to confuse rather than enlighten the reader. For this reason, we give a basic implementation version of Jacobson-Scheidler-Williams NUCOMP as Algorithm 5.10 and a complete implementation is presented in the following chapter.

The version presented here contains improvements relating to the relative generator computation. To see that the new relative generator calculation is correct, observe that the NUCOMP reduction loop embeds partial quotients $[\tilde{q}_0, \tilde{q}_1, \dots, \tilde{q}_i]$ (for some i) into the continued fraction expansion

$$\frac{P_0 + \sqrt{D}}{Q_0} = \left[\tilde{q}_0, \tilde{q}_1, \dots, \tilde{q}_i, \frac{P_{i+1} + \sqrt{D}}{Q_{i+1}} \right],$$

and the Lagrange reduction loop embeds partial quotients $[q_{i+1}, q_{i+2}, \dots, q_{i+j}]$ (for some j) into the continued fraction expansion

$$\frac{P_{i+1} + \sqrt{D}}{Q_{i+1}} = \left[q_{i+1}, q_{i+2}, \dots, q_{i+j}, \frac{P_{i+j+1} + \sqrt{D}}{Q_{i+j+1}} \right].$$

Combining these two we obtain,

$$\frac{P_0 + \sqrt{D}}{Q_0} = \left[\tilde{q}_0, \tilde{q}_1, \dots, \tilde{q}_i, q_{i+1}, q_{i+2}, \dots, q_{i+j}, \frac{P_{i+j+1} + \sqrt{D}}{Q_{i+j+1}} \right].$$

Hence, we may view these two reduction loops as a single loop for the purpose of computing the relative generator. Algorithm 3.4 tells us that all that is required to

compute the relative generator is the sequence $\{B_i\}$. Specifically, the formula is

$$(Q_0, P_0) = \left(B_{i+j-1} + B_{i+j} \left(\frac{P_{i+j+1} - \sqrt{D}}{Q_{i+j+1}} \right) \right) (Q_{i+j+1}, P_{i+j+1}) .$$

The $\{B_i\}$ computed in Algorithm 3.4 are in fact the absolute values of the $\{d_i\}$ computed in the NUCOMP algorithm. The advantages of computing the relative generator in this new manner are:

1. The calculations of G and G' are not required.
2. Storing the q_i in the Lagrange reduction loop is not required.
3. Computing the sequence $\{T_i\}$ is traded for computing the extra $\{B_i\}$ whose magnitude is much smaller.

5.7 Complexity

There has been surprisingly little study done to provide insight into the time and space complexity of reduction algorithms. The most complete study the author is aware of was done by Biehl and Buchmann in [BB97]. Here they improve upon Lagarias's proof in [Lag80] that classical reduction is bounded by $O(n(M(n)))$ where n in our case can be considered to be the bit length of the discriminant and $M(n)$ is the time required to multiply two n -bit integers. Classical multiplication algorithms have running time $O(n^2)$ and Schönhage's multiplication algorithm is asymptotically better with running time $O(n \log n \log \log n)$. Hence, reduction may be performed in time $O(n^3)$ using classical multiplication and $O(n^2 \log n \log \log n)$ using Schönhage's multiplication algorithm.

Algorithm 5.10 Jacobson-Scheidler-Williams NUCOMP (Basic)

Input: Two reduced ideals (Q_a, P_a) and (Q_b, P_b) and also D **Output:** A reduced product (Q, P) and relative generator Ψ where
 $(\Psi)(Q_a, P_a)(Q_b, P_b) = (Q, P)$

[Initialization]

Find S, X, Y, Z so $\sigma S = \gcd(Q_a, Q_b, P_a + P_b) = XQ_a + YQ_b + Z(P_a + P_b)$ $R_b \leftarrow (P_b^2 - D)/Q_b$ $U \leftarrow Y(P_b - P_a) - ZR_b \pmod{Q_a/S}$, (with $0 \leq U < Q_a/S$) $a_{-1} \leftarrow Q_a/S$, $a_0 \leftarrow U$ $d_{-1} = 0$, $d_0 \leftarrow 1$ (Note: $\{|d_i|\}$ are the usual convergents $\{B_i\}$) $i \leftarrow 0$ [Continued fraction expansion of $U/(Q_a/S)$]**while** $a_i > \sqrt[4]{D}$ **do** $i \leftarrow i + 1$ $q_i \leftarrow \lfloor a_{i-2}/a_{i-1} \rfloor$, $a_i \leftarrow a_{i-2} - q_i a_{i-1}$ $d_i \leftarrow d_{i-2} - q_i d_{i-1}$ [Compute (Q_{i+1}, P_{i+1})] $b_i \leftarrow \frac{1}{a_{-1}} \left(a_i \frac{Q_b}{\sigma S} + d_i(P_b - P_a) \right)$ $c_i \leftarrow \frac{1}{a_{-1}} (a_i(P_a + P_b) + \sigma d_i S R_b)$ $c_{i-1} \leftarrow \frac{1}{a_{-1}} (a_{i-1}(P_a + P_b) + \sigma d_{i-1} S R_b)$ $Q_{i+1} \leftarrow (-1)^{i+1} (a_i b_i + c_i d_i)$ $P_{i+1} \leftarrow (-1)^{i+1} (a_{i-1} b_i + c_{i-1} d_i) + P_b$

[Final Lagrange Reduction Steps]

 $(Q, P), \Psi \leftarrow \text{Algorithm 3.4 } (Q_{i+1}, P_{i+1})$ where $B_{-2} \leftarrow |d_{i-1}|$ and $B_{-1} \leftarrow |d_i|$ **return** $(Q, P), \Psi$

Biehl and Buchmann improve the combination of classical reduction with classical multiplication to $O(n^2)$. The proof uses the same idea as the well-known proof that the Euclidean GCD of two numbers may be computed in $O(n^2)$. That is, a more careful analysis is possible since the size of the parameters decrease as the reduction

progresses.

Schönhage shows in [Sch91] that his reduction algorithm takes time $O(\log n M(n))$ and so using Schönhage multiplication, his reduction is bounded by $O(n(\log n)^2 \log \log n)$. Unfortunately, the results we have are of little value in the cases of Schönhage reduction with classical multiplication and classical reduction with Schönhage multiplication. This information is reflected in the following table.

Algorithm	Classical Reduction	Schönhage Reduction
Classical Multiplication	$O(n^2)$	$O(n^2 \log n)$
Schönhage Multiplication	$O(n^2 \log n \log \log n)$	$O(n(\log n)^2 \log \log n)$

Table 5.1: Known Reduction Algorithm Complexity Results

It would be highly desirable to know if Schönhage reduction could be proved to be better than $O(n^2)$ using classical multiplication. Similarly, we are interested to know how tight the bound on classical reduction with Schönhage's multiplication could be shown to be.

Chapter 6

Implementation and Timings

6.1 General Principles

In this chapter we give fit for implementation versions of the algorithms introduced in the previous chapter. We also compare most of the algorithms and present timings for various sizes of discriminant. We have not implemented the Schnorr-Seysen algorithm because it is nearly identical to JSW-NUCOMP but will clearly be slower since it requires the computation of the unreduced product. Also, Shanks' original NUCOMP has not been implemented because it is already known to be slower than JSW-NUCOMP.

The algorithms are presented here as pseudo-code and tested using a full C implementation. We start with some general principles that apply to all of the algorithms and should be considered by anyone implementing them.

- Multi-precision floating point operations should be avoided as they are very costly in terms of computation time. Instead, adapt any such calculations to

use integer arithmetic. For example, the floor of a rational number should not be computed using floating point operations but instead using an integer function in the programming language that performs this operation on input of the numerator and denominator of the rational number.

- If the computer hardware represents numbers in binary, multiplications or divisions by 2^k for $k \in \mathbb{Z}$ should be implemented using bit shifts.
- With packages such as the GNU multi-precision library (GMP), a specific function exists for exact integer divisions. This function should be used whenever possible; for example, the computation of $R = (P^2 - D)/Q$ is an exact division.
- Variables are listed with subscripts but these are only included for elucidation and the programmer must take care to carry only as many variables as is necessary.
- The computation $(-1)^i$ should not be computed directly. If the computation of i is necessary, then examining the least significant bit will indicate whether i is even or odd and $(-1)^i$ is determined with this information. If i is not required, track the value of $(-1)^i$ via a sign variable.
- If algorithms require the value R_0 for efficiency, this is computed in the initialization section and updated along with the Q_i and P_i values throughout. In practice, the programmer should determine if it is beneficial to carry this value between procedures and adjust the algorithms appropriately. In our case of key exchange, there are frequent multiplications and the R value is required for only one of the two ideals in the formulas. Since there is no benefit to computing R

in the algorithm that determines the near reduced ideal, we do not compute R in that algorithm since we would be computing it twice as often as we need to.

- These algorithms assume a fixed field $\mathbb{Q}(\sqrt{D})$; therefore, D , $\lfloor \sqrt{D} \rfloor$ and $\lfloor \sqrt[4]{D} \rfloor$ should be precomputed and available globally to avoid computing this value in each function that requires it.
- When performing calculations involving a multi-precision number and a single-precision number, utilize functions specifically designed for this purpose whenever they are available.
- Statements such as

$$q \leftarrow \left\lfloor \frac{b}{a} \right\rfloor, r \leftarrow b - qa$$

can usually be implemented with commands that return both the quotient and remainder simultaneously. This applies to both single and multi-precision operands.

- Operations with multi-precision variables should take advantage of pointers if they are available. In particular, passing multi-precision variables by value to and from functions should be avoided.

For each individual reduction, the integer coefficients of the relative generator Ψ are manageable; however, the number of reductions involved in computations such as a key exchange soon cause them to grow out of control. As we mentioned in Chapter 4, an exact representation cannot be used because the integer coefficients grow so large they are simply impossible to store. In the C implementation of these

algorithms we have used the (f, p) representation of ideals presented in [JSW] to obtain an approximation to Ψ . For fixed $p \in \mathbb{N}$, $2^k d / 2^p$ is an approximation to Ψ such that

$$\left| \frac{2^p \Psi}{2^k d} - 1 \right| < \frac{f}{2^p} ,$$

where $d, f \in \mathbb{N}$, $k \in \mathbb{Z}$, $2^p < d \leq 2^{p+1}$, and $1 \leq f < 2^p$. That is,

$$\frac{2^k d}{2^p} \left(1 - \frac{f}{2^p} \right) < \Psi < \frac{2^k d}{2^p} \left(1 + \frac{f}{2^p} \right) .$$

A full explanation of (f, p) representations and an analysis of appropriate choices for p may be found in [JSW01] and [JSW]. The Ψ values returned by these algorithms are shown as exact values. When programming, an approximation such as that just described would need to be used.

6.2 Lehmer's Extended GCD Algorithm

Recall from Section 5.1 that Lehmer had a method of speeding up GCD computations. Many of the following algorithms use his idea of taking a single-precision approximation to a multi-precision number, performing computations with the approximation, and then applying a catch-up to the multi-precision number based on the work done on the approximation.

When computing the GCD, Lehmer provides a test which determines when the single-precision steps no longer yield the correct quotients. In [Jeb95], Jebelean simplified the condition tested and it is this improved version that is presented as Algo-

rithm 6.1. For a full discussion of Lehmer's algorithm and Jebelean's condition we refer the reader to Sorenson's analysis given in [Sor95].

Often in our algorithms the GCD is not actually required and it is desirable to exit the GCD algorithm early. If it is sufficient to find numbers A_j and B_j such that $A_j C_{-1} + B_j C_{-2} \leq M$ for some $M \geq \gcd(C_{-2}, C_{-1})$, then the condition **while** $C_{j-1} > 0$ at the first step in the main loop may simply be replaced with the condition **while** $C_{j-2} > M$.

Computing the continued fraction expansion of a rational number involves the same procedure as computing the GCD of the numerator and denominator of that number. Whenever an algorithm requires the GCD of two multi-precision numbers or the continued fraction expansion of a multi-precision rational number, Lehmer's GCD algorithm should be used. Substituting Lehmer's algorithm into the appropriate places is straight-forward and so we have not explicitly included it in the following algorithms to avoid unnecessary clutter.

Algorithm 6.1 Lehmer's Extended GCD (with Jebelean's condition)**Input:** Non-negative integers C_{-2} and C_{-1} with $C_{-2} \geq C_{-1}$ **Output:** (A_{j-1}, B_{j-1}, G) such that $A_{j-1}C_{-1} + B_{j-1}C_{-2} = G = \gcd(C_{-2}, C_{-1})$ **Note:** The variables A_j and B_j correspond in absolute value to the standard sequence from the continued fraction expansion of C_{-2}/C_{-1} .

[Initialization]

 $A_{-2} \leftarrow 0, A_{-1} \leftarrow 1, B_{-2} \leftarrow 1, B_{-1} \leftarrow 0$ $i \leftarrow 0, j \leftarrow 0$

[Main loop]

while $C_{j-1} > 0$ **do** $k \leftarrow \max(\lfloor \log_2 C_{j-2} \rfloor + 1 - (\text{word size}), 0)$ $r_{-2} \leftarrow \lfloor C_{j-2}/2^k \rfloor, r_{-1} \leftarrow \lfloor C_{j-1}/2^k \rfloor$ $a_{-2} \leftarrow 0, a_{-1} \leftarrow 1, b_{-2} \leftarrow 1, b_{-1} \leftarrow 0$

[Euclidean step]

while $r_{i-1} > 0$ **do** {Exit is usually at Jebelean's condition} $q_i \leftarrow \lfloor r_{i-2}/r_{i-1} \rfloor, r_i \leftarrow r_{i-2} - q_i r_{i-1}$ $a_i \leftarrow a_{i-2} - q_i a_{i-1}, b_i \leftarrow b_{i-2} - q_i b_{i-1}$

[Jebelean's condition]

if i is even **then****if** $(r_i < -a_i)$ **or** $(r_{i-1} - r_i < b_i - b_{i-1})$ **then break****else****if** $(r_i < -b_i)$ **or** $(r_{i-1} - r_i < a_i - a_{i-1})$ **then break** $i \leftarrow i + 1$

[Multi-precision step]

if $i = 0$ **then** {No correct single-precision steps} $Q \leftarrow \lfloor C_{j-2}/C_{j-1} \rfloor, C_j \leftarrow C_{j-2} - QC_{j-1}$ $A_j \leftarrow A_{j-2} - QA_{j-1}, B_j \leftarrow B_{j-2} - QB_{j-1}$ **else** $C_{j+i-1} \leftarrow a_{i-2}C_{j-1} + b_{i-2}C_{j-2}, C_{j+i} \leftarrow a_{i-1}C_{j-1} + b_{i-1}C_{j-2}$ $A_{j+i-1} \leftarrow a_{i-2}A_{j-1} + b_{i-2}A_{j-2}, A_{j+i} \leftarrow a_{i-1}A_{j-1} + b_{i-1}A_{j-2}$ $B_{j+i-1} \leftarrow a_{i-2}B_{j-1} + b_{i-2}B_{j-2}, B_{j+i} \leftarrow a_{i-1}B_{j-1} + b_{i-1}B_{j-2}$ $j \leftarrow j + i + 1, i \leftarrow -1$ **return** $(A_{j-1}, B_{j-1}, C_{j-2})$

6.3 Rickert's Algorithm

The implementation of Rickert's algorithm is given as Algorithm 6.2. We could have computed R_{i+2} as $\frac{P_{i+2}^2 - D}{Q_{i+2}}$ instead of the way it is presented; however, this would require a multi-precision squaring and a multi-precision division which are both costly, particularly the division. We instead trade this for three multiplications of a multi-precision number by a single-precision number.

The algorithm uses Algorithm 3.4 to do the single-precision reduction. There is a possibility that the convergents returned by Algorithm 3.4 will produce an overflow when they are multiplied later in Algorithm 6.2. For this reason, it is important to test that the convergents remain less than (word size)/2 in length in Algorithm 3.4 and exit early if necessary.

As discussed in the previous chapter, \widehat{D}_i may be negative. We can ensure it is positive by setting $\widehat{Q}_i = (Q_i/|Q_i|) \lfloor |Q_i|/2^k \rfloor$, $\widehat{P}_i = (P_i/|P_i|) \lfloor |P_i|/2^k \rfloor$ and $\widehat{R}_i = (R_i/|R_i|) \lfloor |R_i|/2^k \rfloor$; then,

$$\widehat{P}_i^2 = \left\lfloor \frac{|P_i|}{2^k} \right\rfloor^2 \geq \left(\frac{P_i}{2^k} \right)^2 > \left(\frac{Q_i}{2^k} \right) \left(\frac{R_i}{2^k} \right) \geq \left(\frac{Q_i}{|Q_i|} \right) \left\lfloor \frac{|Q_i|}{2^k} \right\rfloor \left(\frac{R_i}{|R_i|} \right) \left\lfloor \frac{|R_i|}{2^k} \right\rfloor = \widehat{Q}_i \widehat{R}_i .$$

Here we used the fact that $P^2 - QR = D > 0 \Rightarrow P^2 > QR$. Since $\widehat{P}_i^2 > \widehat{Q}_i \widehat{R}_i$ we also have $\widehat{P}_i^2 - \widehat{Q}_i \widehat{R}_i = \widehat{D}_i > 0$. Another approach would be to take the floor in each case and if $\widehat{D}_i < 0$, simply set $\widehat{Q}_i \leftarrow -\widehat{Q}_i$ and recalculate \widehat{D}_i which will now be positive.

Algorithm 6.2 Rickert–Style Reduction Algorithm (Implementation)**Input:** (Q_0, P_0) , D **Output:** (Q_i, P_i) , Ψ_i such that $(Q_i, P_i) = (\Psi_i)(Q_0, P_0)$

[Initialization]

$$B_{-2} \leftarrow 1, B_{-1} \leftarrow 0, i \leftarrow 0$$

$$R_0 \leftarrow \frac{P_0^2 - D}{Q_0}$$

[Reduction]

while $|Q_i| > \sqrt{D}$ **do**

$$k \leftarrow \max(\max(\lceil \log_2 Q_i \rceil, \lceil \log_2 P_i \rceil, \lceil \log_2 R_i \rceil) - (\text{word size}), 0)$$

$$\hat{Q}_i \leftarrow (Q_i / |Q_i|) \lfloor |Q_i| / 2^k \rfloor, \hat{P}_i \leftarrow (P_i / |P_i|) \lfloor |P_i| / 2^k \rfloor,$$

$$\hat{R}_i \leftarrow (R_i / |R_i|) \lfloor |R_i| / 2^k \rfloor$$

$$\hat{D}_i \leftarrow \hat{P}_i^2 - \hat{Q}_i \hat{R}_i$$

if \hat{Q}_i , \hat{P}_i or \hat{R}_i is 0 **or** $|\hat{Q}_i| < \sqrt{\hat{D}_i}$ **then**

[Full multi-precision reduction]

$$\text{Compute } q_i, r_i \text{ so that } P_i + \lfloor \sqrt{D} \rfloor = q_i Q_i + r_i \quad (0 \leq r_i < |Q_i|)$$

$$P_{i+1} \leftarrow \lfloor \sqrt{D} \rfloor - r_i, Q_{i+1} \leftarrow q_i(P_i - P_{i+1}) - R_i, R_{i+1} \leftarrow -Q_i$$

$$B_{i-1} \leftarrow -B_{i-1}, B_i \leftarrow q_i B_{i-1} - B_{i-2}, i \leftarrow i + 1$$

else

[Single-precision reduction]

$$a, a', b, b' \leftarrow \text{Last two } A_i \text{ and } B_i \text{ from Algorithm 3.4 } ((\hat{Q}_i, \hat{P}_i), \sqrt{\hat{D}_i})$$

$$\varepsilon \leftarrow ab' - ba' \text{ (Compute using Equation (3.12))}$$

$$Q_{i+2} \leftarrow \varepsilon(a^2 Q_i - 2ab P_i + b^2 R_i)$$

$$P_{i+2} \leftarrow \varepsilon(-aa' Q_i + (ab' + a'b) P_i - bb' R_i)$$

$$R_{i+2} \leftarrow \varepsilon(a'^2 Q_i - 2a'b' P_i + b'^2 R_i)$$

$$B_i \leftarrow -a' B_{i-1} + b' B_{i-2}$$

$$B_{i+1} \leftarrow a B_{i-1} - b B_{i-2}$$

$$i \leftarrow i + 2$$

[Compute relative generator]

$$\Psi_i^{-1} \leftarrow B_{i-2} + B_{i-1} \left(\frac{P_i - \sqrt{D}}{Q_i} \right)$$

return (Q_i, P_i) , Ψ_i

6.4 Schönhage's Algorithm

We present the implementations of the various algorithms of Schönhage's method. The transformations are straight-forward except perhaps for the calculations pertaining to Ψ in Algorithm 6.3.

Algorithm 6.3 Schönhage Reduction (Implementation)

Input: $\alpha = (Q, P), D$

Output: $\alpha' = (Q', P'), \Psi$ such that $\alpha' = (\Psi)\alpha$ and α' is reduced

[Initialization]

$$R \leftarrow \frac{P^2 - D}{Q}$$

[Make Positive]

$Q', P', R', c_1, d_1, \text{sign} \leftarrow \text{Algorithm 6.4 } (Q, P, R)$

[Main Reduction]

$Q', P', R', \begin{pmatrix} a_2 & b_2 \\ c_2 & d_2 \end{pmatrix} \leftarrow \text{Algorithm 6.5 } (Q', P', R', -1)$

$c_3 \leftarrow a_2 c_1 + c_2 d_1, d_3 \leftarrow b_2 c_1 + d_1 d_2$

if $\text{sign} = -1$ **then**

$Q' \leftarrow -Q', P' \leftarrow -P', R' \leftarrow -R'$

[Final Lagrange Reduction Steps]

$(Q', P'), \Psi \leftarrow \text{Algorithm 3.4 } ((Q', P'), D)$ where $B_{-2} \leftarrow d_3$ and $B_{-1} \leftarrow -c_3$.

return α', Ψ

The matrix $\begin{pmatrix} a_2 & b_2 \\ c_2 & d_2 \end{pmatrix}$ is returned by the main reduction algorithm, namely Algorithm 6.5. To compute the relative generator we need to multiply this matrix on the left by the matrix returned by Algorithm 5.4. However, by Lemma 3.7 all we require in the Lagrange algorithm that finishes the reduction are the two elements from the second row of this product. The two elements from row 2 are $a_2 c_1 + c_2 d_1$ and $b_2 c_1 + d_1 d_2$ and this calculation only requires c_1 and d_1 from Algorithm 6.4; hence, we only return c_1 and d_1 from the full $\begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix}$ that was computed in the theoretical version of the

"Make Positive" Algorithm 5.4. Initializing B_{-2} and B_{-1} as we have will yield the correct relative generator by Lemma 3.7.

Algorithm 6.3 accepts as input an ideal and returns a reduced equivalent ideal along with its relative generator. The supporting algorithms are listed in the order in which they are called.

Algorithm 6.4 Make Positive (Implementation)

Input: Q, P, R where (Q, P) is an ideal.

Output: $Q', P', R', c, d, \text{sign}$, where $\left(d + c \left(\frac{P' - \sqrt{D}}{Q'} \right)\right) (Q', P') = (Q, P)$ and Q', P', R' are all positive.

[Initialization]

$Q' \leftarrow Q, P' \leftarrow P, R' \leftarrow R$

$c \leftarrow 0, d \leftarrow 1$ {Denotes c and d of Definition 3.4}

[Ensure $Q' + R' \geq 0$ and $P' \geq 0$]

if $(Q' + R' < 0)$ **then**

$Q' \leftarrow -Q', P' \leftarrow -P', R' \leftarrow -R'$

$\text{sign} \leftarrow -1$

else

$\text{sign} \leftarrow 1$

if $(P' < 0)$ **then**

$Q' \leftarrow R', P' \leftarrow -P', R' \leftarrow Q'$

$c \leftarrow -1, d \leftarrow 0$

[Ensure Q', P', R' are all positive]

if $(Q' < 0)$ **then**

$Q' \leftarrow Q' + 2P' + R', P' \leftarrow P' + R', R' \leftarrow R'$

$c \leftarrow c + d, d \leftarrow d$

else if $(R' < 0)$ **then**

$Q' \leftarrow Q', P' \leftarrow Q' + P', R' \leftarrow Q' + 2P' + R'$

$c \leftarrow c, d \leftarrow c + d$

return $Q', P', R', c, d, \text{sign}$

Algorithm 6.5 Monotone Reduction — MR (Implementation)**Input:** Q, P, R, m where (Q, P) is an ideal.**Output:** Q', P', R', N where N is a matrix tracking the relative generator and (Q', P') is minimal above 2^m

[Initialization]

 $Q' \leftarrow Q, P' \leftarrow P, R' \leftarrow R$

[1. If sufficiently small in size, skip to end]

if $\min(Q', 2P', R') < 2^{m+2}$ **then** $N \leftarrow I_2$ **else**

[2. Calculate size of operands and split if necessary]

 $n \leftarrow \max(\lfloor \log_2 Q' \rfloor, \lfloor \log_2 2P' \rfloor, \lfloor \log_2 R' \rfloor) - m + 1$ **if** $m \leq n$ **then** $m' \leftarrow m, p \leftarrow 0$ **else** $m' \leftarrow n, p \leftarrow m - n + 1$ Split Q', P', R' as $Q' = Q_0 + 2^p Q_1$ ($0 \leq Q_0 < 2^p$) $P' = P_0 + 2^p P_1$ ($0 \leq P_0 < 2^p$) $R' = R_0 + 2^p R_1$ ($0 \leq R_0 < 2^p$) $Q' \leftarrow Q_1, P' \leftarrow P_1, R' \leftarrow R_1$

[3. Calculate half-way point]

 $h \leftarrow m' + \lfloor n/2 \rfloor$ **if** $\min(Q', 2P', R') < 2^h$ **then** $N \leftarrow I_2$ **else**[4. Recursive call — reduce (Q, P) minimal above 2^h] $Q', P', R', N \leftarrow \text{MR}(Q', P', R', h)$ [5. Reduce until $\max(Q', 2P', R') < 2^h$]**while** $\max(Q', 2P', R') \geq 2^h$ **do****if** (Q', P') is minimal above $2^{m'}$ **then****break****else** $Q', P', R', N \leftarrow \text{Algorithm 6.6}(Q', P', R', N, m')$ **if** (Q', P') is not minimal above $2^{m'}$ **then**[6-7. Recursive call — reduce (Q', P') minimal above m'] $Q', P', R', N' \leftarrow \text{MR}(Q', P', R', m')$

```

 $N \leftarrow NN'$ 
[ 8. Update tails ]
if  $p > 0$  then
     $F \leftarrow aP_0 - cR_0, G \leftarrow aQ_0 - cP_0$ 
     $Q' \leftarrow 2^p Q' + aG - cF$ 
     $P' \leftarrow 2^p P' - bG + dF$ 
     $R' \leftarrow 2^p R' + b^2 Q_0 + 2bdP_0 + d^2 R_0$ 
[ 9. Reduce above  $m$  ]
while  $(Q', P')$  is not minimal above  $2^m$  do
     $Q', P', R', N \leftarrow \text{Algorithm 6.6 } (Q', P', R', N, m')$ 
return  $Q', P', R', N$ 

```

Algorithm 6.6 Simple Step Above 2^m (Implementation)

Input: Q, P, R, N, m where (Q, P) is an ideal and $N = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

Output: Q', P', R', N' where (Q', P') has been reduced one simple step

$D \leftarrow P^2 - QR$ {Store in a lookup table indexed by Q, P, R }

if $(P > Q)$ then

[Low step]

$t \leftarrow \max(\lceil \sqrt{D + 2^m Q} \rceil, 2^m)$

$q \leftarrow \left\lfloor \frac{P - t}{Q} \right\rfloor$

$Q' \leftarrow Q, P' \leftarrow P - qQ, R' \leftarrow R - q(P + P')$

$N' \leftarrow \begin{pmatrix} a & b - aq \\ c & d - cq \end{pmatrix}$

else

[High step]

$t \leftarrow \max(\lceil \sqrt{D + 2^m R} \rceil, 2^m)$

$q \leftarrow \left\lfloor \frac{P - t}{R} \right\rfloor$

$R' \leftarrow R, P' \leftarrow P - qR, Q' \leftarrow Q - q(P + P')$

$N' \leftarrow \begin{pmatrix} a - bq & b \\ c - dq & d \end{pmatrix}$

return Q', P', R', N'

6.5 A New Algorithm For Ideal Reduction

In this section we present an implementable version of the algorithm introduced in Section 5.4 and discuss some possible modifications to compute R . Given C_{-2} and C_{-1} , by Equation (5.4) the remainders in the continued fraction expansion of C_{-2}/C_{-1} are defined to be

$$C_i = C_{i-2} - \tilde{q}_i C_{i-1} .$$

In Algorithm 6.7, we set $C_{-2} = P_0$ and $C_{-1} = Q_0$ and so Lemma 5.3 tells us that $C_i = (-1)^{i-1}(A_i C_{-1} - B_i C_{-2})$; hence, $G_i = (-1)^{i-1} C_i$. This eliminates the necessity of calculating the $\{A_i\}$.

From the inequality given in Equation (5.11) we observe that the $\{B_i\}$ are bounded in magnitude by $\sqrt[4]{|D|}$. If the continued fraction expansion of P_0/Q_0 is implemented using Lehmer's extended GCD algorithm, most of the calculations in Algorithm 6.7 will involve relatively small numbers.

Unlike all of the other algorithms presented in this thesis, the knowledge of R_0 does not give any advantage in the computations. Equation (3.6) gives the following formulas which utilize R_0 . The formula is also given for the computation of A_i since A_i and A_{i-1} are also required.

$$A_i = \frac{G_i + P_0 B_i}{Q_0}$$

$$F_i = A_i P_0 - B_i R_0$$

$$Q_{i+1} = (-1)^{i+1}(A_i G_i - B_i F_i)$$

$$P_{i+1} = (-1)^{i+1}(-A_{i-1} G_i + B_{i-1} F_i)$$

Algorithm 6.7 Efficient Ideal Reduction (Implementation)**Input:** $\mathfrak{a}_0 = (Q_0, P_0)$, D , where $Q_0, P_0 > 0$ **Output:** Ideal \mathfrak{a}_{i+1} and Ψ_{i+1} such that $\mathfrak{a}_{i+1} = (\Psi_{i+1})\mathfrak{a}_0$ (If $D > 0$, \mathfrak{a}_{i+1} is reduced, if $D < 0$, \mathfrak{a}_{i+1} is within 3 steps of being reduced.)

```

[ Initialization ]
 $i \leftarrow -2$ 
 $B_{-2} \leftarrow 1, B_{-1} \leftarrow 0$ 
 $C_{-2} \leftarrow P_0, C_{-1} \leftarrow Q_0$ 
[ Continued fraction expansion of  $P_0/Q_0$  ]
while  $B_{i+1} < \left\lfloor \sqrt{\lfloor Q_0 / \lfloor \sqrt{|D|} \rfloor} \right\rfloor$  do
     $i \leftarrow i + 1$ 
     $q_{i+1} \leftarrow \lfloor C_{i-1} / C_i \rfloor, C_{i+1} \leftarrow C_{i-1} - q_{i+1}C_i$ 
     $B_{i+1} \leftarrow q_{i+1}B_i + B_{i-1}$ 
[ Compute  $(Q_{i+1}, P_{i+1})$  and  $\Psi_{i+1}$  ]
 $Q_{i+1} \leftarrow (-1)^i \left( \frac{C_i^2 - DB_i^2}{Q_0} \right)$ 
 $P_{i+1} \leftarrow \frac{(-1)^{i-1}C_i - Q_{i+1}B_{i-1}}{B_i}$ 
 $\Psi_{i+1}^{-1} \leftarrow B_{i-1} + B_i \left( \frac{P_{i+1} - \sqrt{D}}{Q_{i+1}} \right)$ 
return  $(Q_{i+1}, P_{i+1}), \Psi_{i+1}$ 

```

$$R_{i+1} = (-1)^{i+1}(-A_{i-1}G_{i-1} + B_{i-1}F_{i-1})$$

We present Table 6.5 which summarizes the calculations by a count of each type of operation.

The size of the denominator in the divisions is important, so note that to calculate Q_i, P_i without R_0 requires a division by Q_0 , which is usually the same size as D when exchanging keys, and a division by B_i which is approximately the same size as $\sqrt[4]{|D|}$. The two divisions utilizing knowledge of R_0 are both by Q_0 . In the case of calculating

	Calculated	Multiplications	Divisions	Additions
Without R_0	Q_i, P_i	4	2	2
Utilizing R_0	Q_i, P_i	8	2	5
Without R_0	Q_i, P_i, R_i	5	3	3
Utilizing R_0	Q_i, P_i, R_i	12	2	7

Table 6.1: Calculation of Q_i, P_i and R_i

R_i without knowledge of R_0 , the extra division is by Q_i which is approximately the same size as $\sqrt{|D|}$.¹

6.6 Jacobson-Scheidler-Williams NUCOMP

In the last chapter we presented a basic version of the Jacobson-Scheidler-Williams NUCOMP algorithm. Here we present a full version with various speedups.

From Shanks' formulas given in Equation (3.1), the GCD computation of three numbers must be done in two stages, so we give the details of the computation. The first GCD calculation is

$$\sigma G = \gcd(Q_a, Q_b) = WQ_a + XQ_b .$$

The W variable is not actually required in the algorithm. Since extended GCD functions are often optimized to benefit from the situation where only one variable is required, the programmer should take advantage of this.

We require on input that $Q_a > Q_b$ and then check if $Q_a < \sqrt[4]{D}$. If this is the case, the product $(Q_a, P_a)(Q_b, P_b)$ is already reduced and we may avoid the reduction

¹The size of Q_0 is discussed in Section 3.3.1 while the sizes of B_i and Q_i come from the proof of Theorem 5.4.

steps.

As noted in their paper, the final reduction may take advantage of Tenner's formulas by computing the previous Q value. Using the subscripts of the algorithm and our notation for R this corresponds to calculating Q_i and setting $R_{i+1} \leftarrow -Q_i$. Since there are only a very few reduction steps needed after the main reduction loop (in practice, from 0 to 3), there is no advantage to using any other method than Lagrange reduction with Tenner's formulas.

Algorithm 6.8 Jacobson-Scheidler-Williams NUCOMP (Implementation)**Input:** Two reduced ideals (Q_a, P_a) and (Q_b, P_b) as well as D with $Q_a \geq Q_b > 0$ **Output:** Reduced (Q, P) and Ψ such that $(\Psi)(Q_a, P_a)(Q_b, P_b) = (Q, P)$

[Initialization]

Find G, W, X so $\sigma G = \gcd(Q_a, Q_b) = WQ_a + XQ_b$ (do not compute W)Find S, Y, Z so $\sigma G = \gcd(G, P_a + P_b) = YG + Z(P_a + P_b)$ $R_b \leftarrow (P_b^2 - D)/Q_b$ $U \leftarrow XY(P_b - P_a) - ZR_b \pmod{Q_a/S}$ (with $0 \leq U < Q_a/S$) $a_{-1} \leftarrow Q_a/S, a_0 \leftarrow U$ $d_{-1} = 0, d_0 \leftarrow 1, i \leftarrow -1$

[If product is reduced, skip to end]

if $Q_a/S \leq \left\lfloor \sqrt[4]{D} \right\rfloor$ then $Q \leftarrow \frac{Q_a Q_b}{\sigma S^2}, P \leftarrow \frac{Q_b U}{\sigma S} + P_b \pmod{Q}, \Psi \leftarrow 1$

else

[Continued fraction expansion of $U/(Q_a/S)$]while $a_{i+1} > \left\lfloor \sqrt[4]{D} \right\rfloor$ do $i \leftarrow i + 1$ $q_i \leftarrow \lfloor a_{i-1}/a_i \rfloor, a_{i+1} \leftarrow a_{i-1} - q_i a_i, d_{i+1} \leftarrow d_{i-1} - q_i d_i$ [Compute $Q_{i+1}, P_{i+1}, R_{i+1}$] $b_i \leftarrow \frac{1}{a_{-1}} \left(a_i \frac{Q_b}{\sigma S} + d_i (P_b - P_a) \right)$ $b_{i-1} \leftarrow \frac{1}{a_{-1}} \left(a_{i-1} \frac{Q_b}{\sigma S} + d_{i-1} (P_b - P_a) \right)$ $c_i \leftarrow \frac{1}{a_{-1}} (a_i (P_a + P_b) + \sigma d_i S R_b)$ $c_{i-1} \leftarrow \frac{1}{a_{-1}} (a_{i-1} (P_a + P_b) + \sigma d_{i-1} S R_b)$ $Q_{i+1} \leftarrow (-1)^{i+1} (a_i b_i + c_i d_i)$ $P_{i+1} \leftarrow (-1)^{i+1} (a_{i-1} b_i + c_{i-1} d_i) + P_b$ $R_{i+1} \leftarrow (-1)^{i+1} (a_{i-1} b_{i-1} + c_{i-1} d_{i-1})$

[Final Lagrange Reduction Steps]

 $(Q, P), \Psi \leftarrow$ Algorithm 3.4 $((Q_{i+1}, P_{i+1}), D)$ where $B_{-2} \leftarrow |d_{i-1}|, B_{-1} \leftarrow |d_i|$, and the R_{i+1} computed above is used.return $(Q, P), \Psi$

6.7 Timings

In this section we present timings for most of the algorithms in this thesis with the intent of discovering the best algorithm to use for each application. We have not implemented the Schnorr-Seysen algorithm because it is the same as JSW-NUCOMP except that it requires the computation of the unreduced product ideal. Thus in an application that requires both multiplication and reduction, it will clearly be slower than JSW-NUCOMP. As well, it cannot be used strictly for reduction since it requires knowledge of the terms Q_a/S and U used in the multiplication. We have also not implemented the original NUCOMP since it is less efficient than the improved version of Section 5.6.

The algorithms were tested by performing a cryptographic key exchange using ideals with different sizes of discriminant. A key exchange involves exponentiating an ideal using the square-and-multiply method; after each squaring or multiplication, the product is reduced and then the near reduced ideal is computed. Hence, this method provides a convenient way of generating a diverse set of ideals for testing reduction.

In all of the key exchanges except NUCOMP, Shanks' formulas given in Equation (3.1) were used for all ideal multiplications. In the case of squaring an ideal, these can be optimized in a straight-forward manner. This eliminates one of the GCD computations and simplifies some of the other equations.

For discriminants with a bit length of 2^k for $k \in \{7, 8, 9, 10, 11, 12, 13\}$, 1000 keys were exchanged. Since we are increasing the bit lengths exponentially, the time involved per key exchange quickly becomes unmanageable. For discriminants with $k \in \{14, 15, 16, 17\}$, we have exchanged just one key but this gives a good indication of the relative performance of the algorithms.

The exponent bound used in the key exchange algorithm is $2^{3 \cdot 2^{k-4}}$ where $k = \lfloor \log_2 \log_2 \Delta \rfloor$ as above. This bound was not chosen to reflect a desired difficulty for an adversary to obtain the key but rather to increase uniformly the number of multiplications and reductions required and thus the work performed by the algorithm. This formula doubles the bit-length of the exponent bound as the bit-length of the discriminant doubles. The constants in the expression were chosen to match the exponent bound of 2^{384} used in [JSW] for discriminants of approximately 2^{3460} .

In that paper they used discriminants ranging in size from 795 to 5704 bits and the corresponding exponent bounds ranged from 160 to 512 bits. Hamdy estimates in [Ham02] that solving the discrete logarithm problem in imaginary quadratic fields for these sizes of discriminant provides 80 to 256 bits of security for cryptographic protocols based on that problem. Vollmer shows in [Vol00] that the best-known algorithm for solving the DLP in real quadratic fields has the same asymptotic complexity as the best-known algorithm that solves the problem in imaginary quadratic fields; hence, these sizes of discriminant should offer at least the same level of security in real quadratic fields (see [Jac00]).

All timings were taken on a computer with the following configuration:

- Platform: IBM x330 Server
- Processor: Dual Pentium IV Xeon 2.8 GHz
- RAM: 1 GB
- Operating System: Red Hat Linux Release 9
- Linux Kernel: 2.4.20-28.9smp

- Compiler: gcc 3.2.2
- Multi-precision Library: GMP 4.1.2

Table 6.2 compares the average number of seconds required per party to exchange one key. At the start of each of the 1000 rounds, a random discriminant of the desired bit length was generated, as well as exponents for the two parties. The initial ideal used was $\rho^5(1)$. Each algorithm was then called in turn to be used in the key exchange. The timings are for the entire key exchange including the multiplication, reduction and near computations. The keys of both parties were compared to ensure they matched. Additionally, the keys returned by each of the algorithms were compared to ensure that each algorithm indeed generated the same key.

k	Bit Size	Gauss	Lagrange	Rickert	New	NUCOMP	Schönhage
7	128	0.0083	0.0023	0.0025	0.0020	0.0018	0.0075
8	256	0.0638	0.0092	0.0091	0.0067	0.0063	0.0323
9	512	0.2487	0.0385	0.0382	0.0250	0.0238	0.1407
10	1024	1.3642	0.1946	0.1815	0.1137	0.1042	0.6114
11	2048	10.3750	1.0936	0.9874	0.6093	0.5489	2.7226
12	4096	64.5476	6.7283	5.9236	3.5253	3.1004	12.5681
13	8192	487.0460	45.1150	38.5300	22.0800	18.4800	61.9850

Table 6.2: Average Key Exchange Times In Seconds

Note that Gaussian reduction is the slowest reduction algorithm by a wide margin. The performance is so slow that in Table 6.3 the reader will notice that it is no longer being compared. Also, the Lagrange and Rickert algorithms are included only until they are clearly slower than Schönhage's Algorithm.

k	Bit Size	Lagrange	Rickert	New	NUCOMP	Schönhage
14	16384	327.24	271.88	149.58	123.72	326.94
15	32768	2429.52	2037.32	1055.73	866.95	1828.28
16	65536	—	—	7503.17	6106.82	10931.36
17	131072	—	—	55829.51	45129.55	68678.16

Table 6.3: Single Key Exchange Times In Seconds

The following two graphs reflect the information contained in Tables 6.2 and 6.3. The graph in Figure 6.1 is based on a subset of the data in Table 6.2 and plots the length of time to exchange a key against the bit size of the discriminant. The graph in Figure 6.2 is a logarithmic graph which helps to make clear the complexity of the various algorithms.

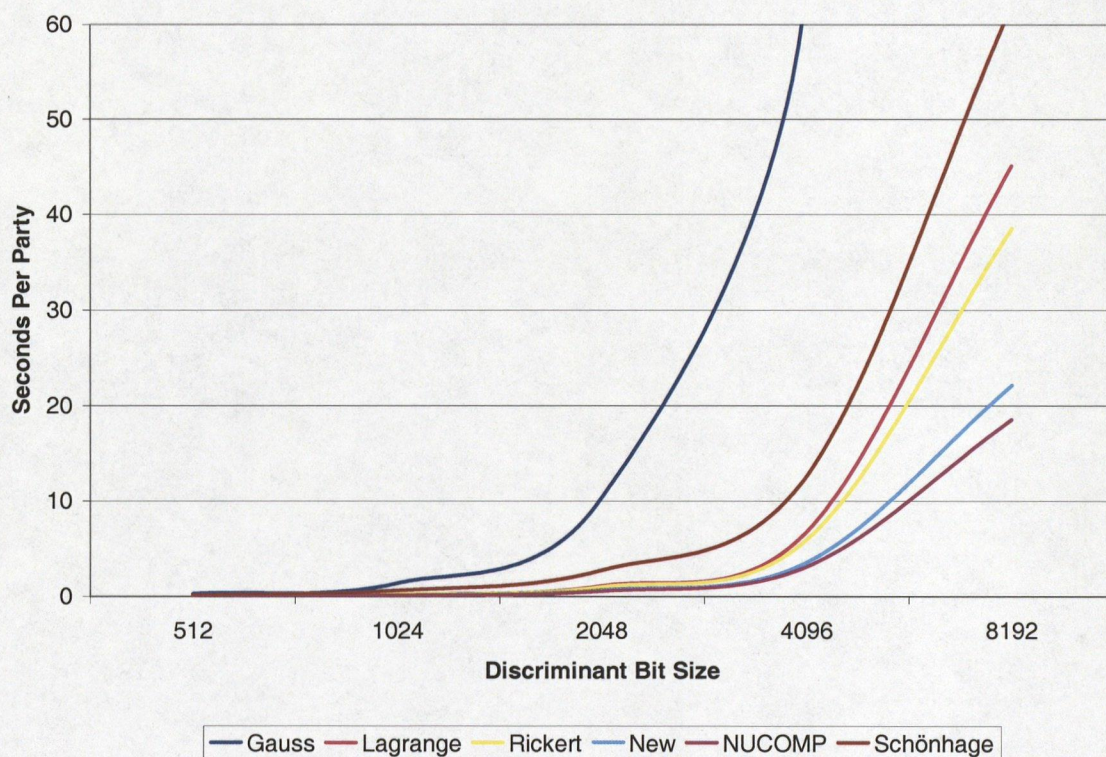


Figure 6.1: Key Exchange Comparison

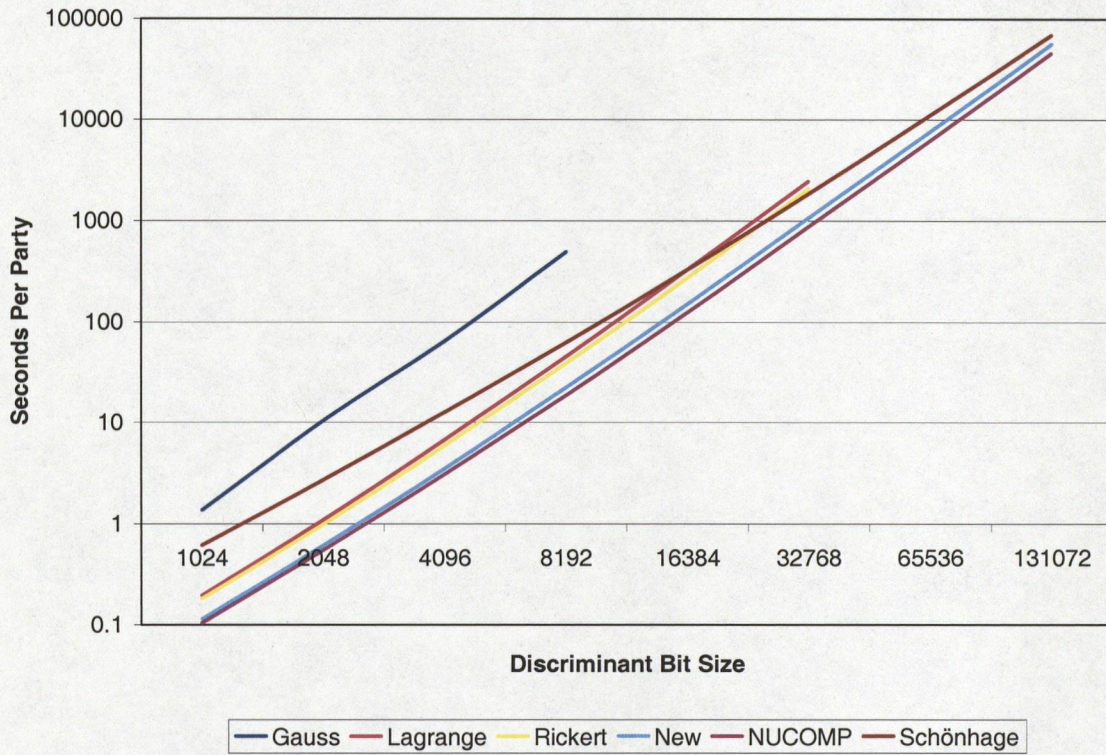


Figure 6.2: Key Exchange Comparison (Logarithmic)

From the graph in Figure 6.2 it is clear that Schönhage's algorithm will eventually be faster than all of the other algorithms. Microsoft Excel's least squares trendline provides the equations in Table 6.4 which give an estimate of the time in seconds that each algorithm will take to exchange a key. The first column of functions is in terms of the bit length of Δ and the second is in terms of $\log_2 \log_2 \Delta$.

Algorithm	$\log_2 \Delta$	$\log_2 \log_2 \Delta$
Gauss	$4.9255 \times 10^{-9} (\log_2 \Delta)^{2.8076}$	$4.9255 \times 10^{-9} e^{1.9461 \log_2 \log_2 \Delta}$
Lagrange	$1.0527 \times 10^{-9} (\log_2 \Delta)^{2.7274}$	$1.0527 \times 10^{-9} e^{1.8905 \log_2 \log_2 \Delta}$
Rickert	$1.2268 \times 10^{-9} (\log_2 \Delta)^{2.6939}$	$1.2268 \times 10^{-9} e^{1.8673 \log_2 \log_2 \Delta}$
New	$1.1395 \times 10^{-9} (\log_2 \Delta)^{2.6391}$	$1.1395 \times 10^{-9} e^{1.8293 \log_2 \log_2 \Delta}$
NUCOMP	$1.3400 \times 10^{-9} (\log_2 \Delta)^{2.6039}$	$1.3400 \times 10^{-9} e^{1.8049 \log_2 \log_2 \Delta}$
Schönhage	$68.656 \times 10^{-9} (\log_2 \Delta)^{2.3073}$	$68.656 \times 10^{-9} e^{1.5993 \log_2 \log_2 \Delta}$

Table 6.4: Estimating Functions For Key Exchange Times

Based on these estimating functions, we may obtain an approximation for when we expect Schönhage's algorithm to overtake NUCOMP.

$$\begin{aligned}
& 68.656 \times 10^{-9} e^{1.5993 \log_2 \log_2 \Delta} = 1.3400 \times 10^{-9} e^{1.8049 \log_2 \log_2 \Delta} \\
\Rightarrow & \frac{68.656}{1.3400} = e^{(1.8049 - 1.5993) \log_2 \log_2 \Delta} \\
\Rightarrow & \log_2 \log_2 \Delta = \frac{\ln 51.236}{0.2056} \approx 19.146 \\
\Rightarrow & \log_2 \Delta \approx 2^{19.146} .
\end{aligned}$$

In other words, we would only expect to benefit from using Schönhage's algorithm once Δ is approximately 580,000 binary digits long. As a decimal number, $\Delta \approx 10^{175,000}$. This is an extremely large number which indicates that Schönhage's algorithm would rarely be used in practice.

For practical applications, JSW-NUCOMP is the fastest method available for all sizes of discriminant when both multiplication and reduction is required. From the data given, we may extrapolate that the new reduction algorithm is the best method to use for all sizes of discriminant when only reduction is required. We reach this conclusion by noticing that the timings include the three steps of multiplication,

reduction and finding the near reduced ideal. The multiplication step in all of the reduction-only timings is exactly the same and so any differences in performance are completely due to the time involved in reducing the ideal and finding the near reduced ideal. Hence, in applications such as solving norm equations, computing the ideal class group, and computing the ideal class number, the new reduction algorithm is the optimal choice.

Chapter 7

Conclusion

In this thesis we reviewed all of the known reduction algorithms for ideals. With the exception of the Jacobson-Scheidler-Williams NUCOMP, this is the first time the modern algorithms have been presented in this language. It is also the first time that the algorithms of Gauss, Rickert, Schönhage, and Schnorr-Seysen have been adapted to ideals of positive discriminant. We provided an easy method of calculating the relative generator from any type of reduction algorithm and provided many example implementations using this method.

We saw that the Schnorr-Seysen algorithm anticipated the new algorithm of Section 5.4 and the improvements to NUCOMP presented in Section 5.6. All of the algorithms with the exception of the Schnorr-Seysen and Shanks algorithms were implemented and compared. The Schnorr-Seysen algorithm was not implemented because the technique is nearly identical to Jacobson-Scheidler-Williams NUCOMP except that efficient formulas for multiplication are not used. Shanks' NUCOMP was not implemented because it is clearly less efficient than the modern NUCOMP

algorithm and our goal is to find the best algorithm to use in each situation.

We saw the similarity of the algorithms in Sections 5.3, 5.4 and 5.6. They all embed the quotients of an approximation to $(P_0 + \sqrt{D})/Q_0$ into the continued fraction expansion of $(P_0 + \sqrt{D})/Q_0$ but the logic by which they were discovered was radically different.

For practical purposes, NUCOMP is the best algorithm to use when multiplication and reduction are both required. Through the work of Shanks, Atkin, van der Poorten, Jacobson, Scheidler and Williams we have the ability to compute a reduced product without actually computing with any parameters near the size of the unreduced product. If all that is required is to reduce an ideal, the new reduction algorithm presented here is the winner according to the data.

Schönhage's algorithm is asymptotically the fastest and is the only algorithm incorporating high-efficiency design techniques with its divide-and-conquer strategy. In the future we may see other strategies used, such as the greedy method, dynamic programming or randomization, which could produce highly efficient algorithms. As we saw, sometimes the asymptotic efficiency of an algorithm does not translate into something which is useful in practice.

Further work which may be done in this area includes:

- Using the new algorithm in place of Lagrange reduction in the single-precision reductions of Rickert's algorithm.
- Incorporating Lehmer-style single precision calculations into Schönhage's algorithm. For example, the while loops and their simple steps may benefit from this.

- Further practical efficiencies may be derived in Schönage's algorithm from performing some separate action once the parameters get close to the base case.
- The key exchange may be made more efficient by using better exponentiation techniques such as those described in [MvOV97]. For example, windowing trades off some precomputation in exchange for fewer and faster calculations during the main exponentiation loop.
- After reducing an ideal with any of the reduction algorithms, Algorithm 3.5 is called to find a near reduced ideal. Andreas Stein at the University of Illinois (Urbana-Champaign) is considering the possibility of adapting the reduction procedure to eliminate or reduce the number of steps required to find a near reduced ideal.
- The algorithms utilizing continued fractions may be able to benefit from Shanks' Baby-Step Giant-Step method to take giant steps toward a near reduced ideal.
- Adapting these techniques to higher degree number fields and function fields.
- Comparing the reduction algorithms across a wide range of sizes of Q and P for a fixed D .

Undoubtedly, it is clear that there is an abundance of work to be done in this area of research. Advancements will be immediately applicable to diverse areas such as solving the Pell equation, cryptography, and calculating the fundamental unit or regulator of a quadratic field.

Bibliography

- [Atk] A.O.L. Atkin, *Letter to D. Shanks on the programs NUDUPL and NU-COMP, 12 December 1988*, From the Nachlass of D. Shanks, made available by Hugh C. Williams.
- [BB97] I. Biehl and J.A. Buchmann, *An analysis of the reduction algorithms for binary quadratic forms*, 1997.
- [BBT94] I. Biehl, J.A. Buchmann, and C. Thiel, *Cryptographic protocols based on discrete logarithms in real-quadratic orders*, Advances In Cryptology – CRYPTO '94 (Heidelberg), vol. Lecture Notes in Computer Science, Springer, 1994, pp. 56–60.
- [BDW90] J.A. Buchmann, S. Düllmann, and H.C. Williams, *On the complexity and efficiency of a new key exchange system*, Advances In Cryptology—EUROCRYPT '89 (Houthalen, 1989), Lecture Notes in Comput. Sci., vol. 434, Springer, Berlin, 1990, pp. 597–616. MR 92a:11150
- [BH01] J.A. Buchmann and S. Hamdy, *A survey on IQ cryptography*, Public-key Cryptography And Computational Number Theory (Warsaw, 2000), de Gruyter, Berlin, 2001, pp. 1–15. MR 2003e:94064

- [Buc03] J.A. Buchmann, *Algorithms for binary quadratic forms*, Internet preprint, www.cdc.informatik.tu-darmstadt.de/~buchmann/AlgorithmsForQuadraticForms.ps, 2003.
- [Bue89] D.A. Buell, *Binary Quadratic Forms*, Springer-Verlag, New York, 1989, Classical theory and modern computations. MR 92b:11021
- [BW88a] J.A. Buchmann and H.C. Williams, *A key-exchange system based on imaginary quadratic fields*, J. Cryptology **1** (1988), no. 2, 107–118. MR 90g:11166
- [BW88b] ———, *On the infrastructure of the principal ideal class of an algebraic number field of unit rank one*, Math. Comp. **50** (1988), no. 182, 569–579. MR 89g:11098
- [BW90] ———, *A key exchange system based on real quadratic fields (extended abstract)*, Advances In Cryptology—CRYPTO '89 (Santa Barbara, CA, 1989), Lecture Notes in Comput. Sci., vol. 435, Springer, New York, 1990, pp. 335–343. MR 91f:94014
- [Coh93] H. Cohen, *A Course In Computational Algebraic Number Theory*, Graduate Texts in Mathematics, vol. 138, Springer-Verlag, Berlin, 1993. MR 94i:11105
- [DH76] W. Diffie and M.E. Hellman, *New directions in cryptography*, IEEE Trans. Information Theory **IT-22** (1976), no. 6, 644–654. MR 55 #10141
- [ElG85] T. ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, Advances In Cryptology (Santa Barbara, Calif.,

- 1984), Lecture Notes in Comput. Sci., vol. 196, Springer, Berlin, 1985, pp. 10–18. MR 87b:94037
- [Fra98] J.B. Fraleigh, *A First Course In Abstract Algebra*, sixth ed., Addison-Wesley Publishing Co., Don Mills, Ont., 1998.
- [Ham02] S. Hamdy, *Über die sicherheit und effizienz kryptographischer verfahren mit klassengruppen imaginär-quadratischer zahlkörper*, Ph.D. thesis, Technische Universität, Darmstadt (Germany), 2002.
- [Hun80] T.W. Hungerford, *Algebra*, Graduate Texts in Mathematics, vol. 73, Springer-Verlag, New York, 1980, Reprint of the 1974 original. MR 82a:00006
- [Jac00] M.J. Jacobson, Jr., *Computing discrete logarithms in quadratic orders*, J. Cryptology **13** (2000), no. 4, 473–492. MR 2003b:94046
- [Jeb95] T. Jebelean, *A double-digit Lehmer-Euclid algorithm for finding the GCD of long integers*, J. Symbolic Comput. **19** (1995), no. 1-3, 145–157, Design and implementation of symbolic computation systems (Gmunden, 1993). MR 96h:11128
- [JSW] M.J. Jacobson, Jr., R. Scheidler, and H.C. Williams, *An improved real quadratic field based key exchange procedure*, Submitted.
- [JSW01] ———, *The efficiency and security of a real quadratic field based key exchange protocol*, Public-key Cryptography And Computational Number Theory (Warsaw, 2000), de Gruyter, Berlin, 2001, pp. 89–112. MR 2003f:94062

- [JvdP02] M.J. Jacobson, Jr. and A.J. van der Poorten, *Computational aspects of NUCOMP*, Algorithmic Number Theory: 5th International Symposium, ANTS-V, Sydney, Australia, July 7-12, 2002. Proceedings (New York) (C. Fieker and D.R. Kohel, eds.), Lecture Notes in Computer Science, vol. 2369, Springer, 2002, pp. 120–133.
- [KW90] P. Kaplan and K.S. Williams, *The distance between ideals in the orders of a real quadratic field*, Enseign. Math. (2) **36** (1990), no. 3-4, 321–358. MR 92e:11028
- [Lag80] J.C. Lagarias, *Worst-case complexity bounds for algorithms in the theory of integral quadratic forms*, J. Algorithms **1** (1980), 142–186.
- [Leh38] D.H. Lehmer, *Euclid's algorithm for large numbers*, The American Mathematical Monthly **45** (1938), no. 4, 227–233.
- [Len82] H.W. Lenstra, Jr., *On the calculation of regulators and class numbers of quadratic fields*, Number Theory Days, 1980 (Exeter, 1980), London Math. Soc. Lecture Note Ser., vol. 56, Cambridge Univ. Press, Cambridge, 1982, pp. 123–150. MR 86g:11080
- [MvOV97] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook Of Applied Cryptography*, CRC Press Series on Discrete Mathematics and its Applications, CRC Press, Boca Raton, FL, 1997, With a foreword by Ronald L. Rivest. MR 99g:94015

- [Ric89] N.W. Rickert, *Efficient reduction of quadratic forms*, Computers And Mathematics (Cambridge, MA, 1989), Springer, New York, 1989, pp. 135–139. MR 90f:11049
- [Ros00] K.H. Rosen, *Elementary Number Theory And Its Applications*, fourth ed., Addison-Wesley, Reading, MA, 2000. MR 2000i:11001
- [SBW94] R. Scheidler, J.A. Buchmann, and H.C. Williams, *A key-exchange protocol using real quadratic fields*, J. Cryptology 7 (1994), no. 3, 171–199. MR 96e:94015
- [Sch82] R.J. Schoof, *Quadratic fields and factorization*, Computational Methods In Number Theory, Part II, Math. Centre Tracts, vol. 155, Math. Centrum, Amsterdam, 1982, pp. 235–286. MR 85g:11118b
- [Sch91] A. Schönhage, *Fast reduction and composition of binary quadratic forms*, International Symposium On Symbolic And Algebraic Computation (ISSAC), ACM Press, 1991, pp. 128–133.
- [Sch01] R. Scheidler, *Cryptography in quadratic function fields*, Des. Codes Cryptogr. 22 (2001), no. 3, 239–264. MR 2001m:94049
- [Sha71] D. Shanks, *Class number, a theory of factorization, and genera*, 1969 Number Theory Institute (Proc. Sympos. Pure Math., Vol. XX, State Univ. New York, Stony Brook, N.Y., 1969), Amer. Math. Soc., Providence, R.I., 1971, pp. 415–440. MR 47 #4932
- [Sha72] ———, *The infrastructure of a real quadratic field and its applications*, Proceedings Of The Number Theory Conference (Univ. Colorado, Boul-

- der, Colo., 1972) (Boulder, Colo.), Univ. Colorado, 1972, pp. 217–224.
MR 52 #10672
- [Sha78] ———, *A matrix underlying the composition of quadratic forms and its implications for cubic extensions*, Notices Amer. Math. Soc. **25** (1978), A305.
- [Sha89] ———, *On Gauss and composition. I, II*, Number Theory And Applications (Banff, AB, 1988), NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci., vol. 265, Kluwer Acad. Publ., Dordrecht, 1989, pp. 163–178, 179–204.
MR 92e:11150
- [Sor95] J. Sorenson, *An analysis of Lehmer's Euclidean GCD algorithm*, Proceedings Of The 1995 International Symposium On Symbolic And Algebraic Computation (New York, NY, USA), ACM Press, 1995, ISBN:0-89791-699-9, pp. 254 – 258.
- [ST87] I. Stewart and D. Tall, *Algebraic Number Theory*, second ed., Chapman and Hall Mathematics Series, Chapman & Hall, London, 1987. MR 88d:11001
- [vdP03] A.J. van der Poorten, *A note on NUCOMP*, Math. Comp. **72** (2003), no. 244, 1935–1946 (electronic). MR 1 986 813
- [vtW01] A.J. van der Poorten, H.J.J. te Riele, and H.C. Williams, *Computer verification of the Ankeny-Artin-Chowla conjecture for all primes less than 100 000 000 000*, Math. Comp. **70** (2001), no. 235, 1311–1328. MR 2001j:11125

- [Vol00] U. Vollmer, *Asymptotically fast discrete logarithms in quadratic number fields*, Algorithmic Number Theory (Leiden, 2000), Lecture Notes in Comput. Sci., vol. 1838, Springer, Berlin, 2000, pp. 581–594. MR 2003b:11135
- [WW87] H.C. Williams and M.C. Wunderlich, *On the parallel generation of the residues for the continued fraction factoring algorithm*, Math. Comp. 48 (1987), no. 177, 405–423. MR 88i:11099

Appendix A

An Improved Composition Algorithm

The unpublished paper, *An Improved Composition Algorithm* by C.P. Schnorr and M. Seysen, is reprinted here with permission.

An Improved Composition Algorithm

C.P. Schnorr and M. Seysen

Fachbereich Mathematik
Universität Frankfurt

August 1983

Abstract We propose an improved composition algorithm for binary quadratic forms with negative discriminant Δ . Given two reduced forms (A_i, B_i, C_i) , $i = 1, 2$, we compute a representative (A_3, B_3, C_3) of the product class by Shanks's formulae. We describe a novel way to reduce (A_3, B_3, C_3) by adapting the Euclidean algorithm to the particular structure of (A_3, B_3, C_3) . The new reduction method takes about the same number of arithmetical operations as the Gauss-reduction, but it works on integers $\leq \sqrt{|\Delta|}/3$ whereas Gauss-reduction operates on integers of order $|\Delta|$. Hence, if long integer arithmetic is implemented by standard software, then the new reduction method for composition is about 4 times faster than Gauss-reduction.

This research has been done under BMFT-grant 08'3108.

1. Introduction and Notations

We consider binary quadratic forms $AX^2 + BXY + CY^2$, $A, B, C \in \mathbb{Z}$, notation (A, B, C) . Our forms shall be positive, i.e. $A, C > 0$ and primitive, i.e. $\gcd(A, B, C) = 1$, and will have negative discriminant $\Delta = B^2 - 4AC < 0$.

Forms are written in matrix notation

$$AX^2 + BXY + CY^2 = (x, y) \begin{pmatrix} A & B/2 \\ B/2 & C \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

Two forms (A, B, C) , (A', B', C') are $SL_2(\mathbb{Z})$ -equivalent if there exists a $(2, 2)$ integer matrix M with $\det M = 1$ (i.e. $M \in SL_2(\mathbb{Z})$) such that (let M^T be the transpose of M):

$$M^T \begin{pmatrix} A & B/2 \\ B/2 & C \end{pmatrix} M = \begin{pmatrix} A' & B'/2 \\ B'/2 & C' \end{pmatrix}.$$

Since $SL_2(\mathbb{Z})$ is a group this yields an equivalence relation.

According to Gauss the $SL_2(\mathbb{Z})$ -equivalence classes of forms with discriminant Δ form an abelian group $G(\Delta)$. The group operation is called composition. Efficient routines for composition are important for the calculations in $G(\Delta)$ and in particular for the factoring algorithms of Shanks (1971) and Schnorr, Lenstra (1982).

$SL_2(\mathbb{Z})$ -equivalence classes of forms are represented by reduced forms.

(A, B, C) is reduced if $|B| \leq A \leq C$. An $SL_2(\mathbb{Z})$ -equivalence class H either contains exactly one reduced form or two reduced forms $(A, \pm B, C)$. In the latter case H is ambiguous, i.e. $H^2 = 1$. Every reduced form satisfies $A \leq \sqrt{|\Delta/3|}$ and $AC \leq |\Delta/3|$.

Given two reduced forms (A_i, B_i, C_i) , $i = 1, 2$, with the same discriminant Δ composition is done in two stages.

A representative (A_3, B_3, C_3) of the product class is computed by Shanks's formulae. We give a novel algorithm for reducing (A_3, B_3, C_3) .

2. The Shanks's formulae

A form (A_3, B_3, C_3) of the product class of two reduced forms (A_i, B_i, C_i) , $i = 1, 2$, with the same discriminant Δ , is given as follows (see Lenstra (1982), p. 127 f.)

$$d := \gcd(A_1, A_2, (B_1 + B_2)/2)$$

Compute $\lambda, \mu, \nu \in \mathbb{Z}$ with

$$d = \lambda A_1 + \lambda A_2 + \nu (B_1 + B_2)/2$$

$$R := (\lambda(B_1 - B_2)/2 - \nu C_2) \bmod (A_1/d) \quad ; \quad 0 \leq R < \frac{A_1}{d}$$

$$A_3 := A_1 A_2 / d^2$$

$$B_3 := B_2 + 2 A_2 R / d$$

$$C_3 := (B_3^2 - \Delta) / (4 A_3)$$

In order to compute λ, μ, ν one first applies the extended Euclidean algorithm (see Knuth (1981), algorithm X, 4.5.2) to A_1, A_2 . This yields $\bar{\lambda}, \bar{\mu}$ with $\gcd(A_1, A_2) = \bar{\mu} A_1 + \bar{\lambda} A_2$. A second application of the extended Euclidean algorithm to $\gcd(A_1, A_2)$ and $(B_1 + B_2)/2$ yields λ', μ' with $d = \mu' \gcd(A_1, A_2) + \nu' (B_1 + B_2)/2$. Hence $\lambda = \mu' \bar{\lambda}$, $\mu = \mu' \bar{\mu}$, $\nu = \nu'$.

In general $\gcd(A_1, A_2)$ will be rather small, therefore the cost of the Euclidean algorithm applied to $\gcd(A_1, A_2)$ and $(B_1 + B_2)/2$ can be neglected.

Our procedure for reducing (A_3, B_3, C_3) in section 3 is based on the following idea.

Reducing the form (A_3, B_3, C_3) means to construct some $M \in \text{SL}_2(\mathbb{Z})$ such that

A'_3 in

$$\begin{pmatrix} A'_3 & B'_3/2 \\ B'_3/2 & C'_3 \end{pmatrix} = M^T \begin{pmatrix} A_3 & B_3/2 \\ B_3/2 & C_3 \end{pmatrix} M$$

becomes minimal. With $M = \begin{pmatrix} p & q \\ r & s \end{pmatrix}$ we have

$$A'_3 = \frac{d}{A_1} \left[\frac{A_2}{d} \left(\frac{A_1}{d} p + Rr \right)^2 + r B_2 \left(\frac{A_1}{d} p + Rr \right) + r^2 C_2 d \right]$$

which has the dominant term $\frac{A_2}{A_1} \left(\frac{A_1}{d} p + Rr \right)^2$. We minimize A'_3 by minimizing $\frac{A_1}{d} p + Rr$ which can be done by applying the extended Euclidean algorithm to A_1/d and R .

3. The new reduction algorithm

W.l.o.g. let $A_2 \leq A_1$ (otherwise interchange (A_1, B_1, C_1) with (A_2, B_2, C_2)).

0) if $A_1 A_2 / d^2 \leq \sqrt{|\Delta|}/3$ then put $(A'_3, B'_3, C'_3) := (A, B, C)$ and stop.

(Application of the extended Euclidean algorithm to A_1/d and R .)

$$1) \begin{pmatrix} u_1^0 & u_2^0 & u_3^0 \\ v_1^0 & v_2^0 & v_3^0 \end{pmatrix} := \begin{pmatrix} 1 & 0 & A_1/d \\ 0 & 1 & R \end{pmatrix} ; \quad i := 0$$

$$2) \text{ while } v_3^i > \sqrt{\frac{A_1}{A_2}} \sqrt{|\Delta|}/3 \text{ do } \left[\begin{pmatrix} u_1^{i+1} & u_2^{i+1} & u_3^{i+1} \\ v_1^{i+1} & v_2^{i+1} & v_3^{i+1} \end{pmatrix} := \begin{pmatrix} 0 & 1 & \\ 1 & -\lfloor u_3^i / v_3^i \rfloor & \end{pmatrix} \begin{pmatrix} u_1^i & u_2^i & u_3^i \\ v_1^i & v_2^i & v_3^i \end{pmatrix} ; i := i+1 \right]$$

$$3) M_1 := \begin{pmatrix} v_1^1 & v_2^1 \\ (-1)^1 u_1^1 & (-1)^1 u_2^1 \end{pmatrix}$$

$$4) \begin{pmatrix} A'_3 & B'_3/2 \\ B'_3/2 & C'_3 \end{pmatrix} = M_1^T \begin{pmatrix} A_3 & B_3/2 \\ B_3/2 & C_3 \end{pmatrix} M_1$$

It easily follows from $\det \begin{pmatrix} 0 & 1 \\ 1 & q_i \end{pmatrix} = -1$ that $\det M_1 = 1$. Hence (A'_3, B'_3, C'_3) is $SL_2(\mathbb{Z})$ -equivalent to (A_3, B_3, C_3) . The next theorems show that a reduced form is obtained from (A'_3, B'_3, C'_3) with only a few arithmetical steps.

Theorem 1 $A'_3 \leq \sqrt{3|\Delta|}$.

Theorem 2 Let (A, B, C) be a form satisfying $|B| < A < 2.5 \sqrt{|\Delta|}$.

Put $\begin{pmatrix} \bar{A} & \bar{B}/2 \\ \bar{B}/2 & \bar{C} \end{pmatrix} := \begin{pmatrix} 1 & 0 \\ \lambda & 1 \end{pmatrix}^T \begin{pmatrix} A & B/2 \\ B/2 & C \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \lambda & 1 \end{pmatrix}$ with $\lambda \in \mathbb{Z}$ such that

$\bar{B} = B + 2\lambda C$, $\bar{C} = C$ satisfies $|\bar{B}| \leq \bar{C}$. Then $|\bar{B}| \leq \bar{A}$.

(Hence either $(\bar{A}, \bar{B}, \bar{C})$ or $(\bar{C}, -\bar{B}, \bar{A})$ is reduced and $SL_2(\mathbb{Z})$ -equivalent to (A, B, C) .)

4. The efficiency of the reduction algorithm

We compare our algorithm to the Gauss-reduction which, applied to (A_3, B_3, C_3) , does the following:

```

while  $(A_3, B_3, C_3)$  is not reduced do
   $B := B_3 - \lfloor B_3/C_3 \rfloor C_3$ 
   $(A_3, B_3, C_3) := (C_3, B, (B^2 - \Delta)/(4C_3))$ 

```

In practise the Gauss-reduction generates almost the same sequence of forms as the algorithm described in section 3. Let (A_3^i, B_3^i, C_3^i) be the form after the i -th Gauss-reduction step. Then in general, with some rare exceptions:

$$\begin{pmatrix} A_3^i & B_3^i/2 \\ B_3^i/2 & C_3^i \end{pmatrix} = M_i^T \begin{pmatrix} A_3^i & B_3^i/2 \\ B_3^i/2 & C_3^i \end{pmatrix} M_i$$

A closer inspection shows that the number of arithmetical steps is almost the same for both reduction algorithms. Nevertheless, algorithm 3 is more efficient since it operates on integers of half the binary length.

Call an integer with absolute value $\leq \sqrt{|\Delta|}$ ($|\Delta|$, $|\Delta|^{3/2}$, resp.) single precision (double precision, triple precision, resp.).

The loop, step2, of the new reduction algorithm operates with single precision integers whereas Gauss-reduction operates with double precision integers. Step 4 and the final reduction, see theorem 2, require at most 12 multiplications some of which have double and triple precision arguments. In case that the arithmetic on long integers is implemented in software the loop (i.e. step 2) of the new reduction algorithm is about 4 times faster than Gauss-reduction. For large integers this by far dominates the additional costs of the at most 12 multiplications on double and triple precision integers at the end. We have extensively tested the new reduction method, and we have used it for an efficient implementation of the integer factoring algorithm of Schnorr and Lenstra (1984).

Summing up the cost for the composition of two equivalence classes of $G(\Delta)$ represented by reduced forms, we obtain

Corollary 3 The average cost for composition in $G(\Delta)$ is essentially the cost of 1.5 applications of the extended Euclidean algorithm to integers of order $\sqrt{|\Delta|}$.

Proof By Shanks's formulae A_3, B_3, C_3 can essentially be computed (neglecting a fixed number of arithmetical steps) at the cost of computing $\gcd(A_1, A_2)$ and $\gcd(\gcd(A_1, A_2), (B_1 + B_2)/2)$ via the extended Euclidean algorithm. For random $x, y \leq n$ the expected value of $\gcd(x, y)$ is about $\frac{6}{\pi^2} \ln n$ which shows that the average cost of computing $\gcd(\gcd(A_1, A_2), (B_1 + B_2)/2)$ can be neglected: in fact $\sum_{i \in \mathbb{Z}} 1/i^2 = \pi^2/6$ implies that $\gcd(x, y) = i$ occurs with probability about $\frac{6}{\pi^2} \frac{1}{i^2}$. Therefore the expected value of $\gcd(x, y)$ for random $x, y \leq n$ is about $\frac{6}{\pi^2} \sum_{i=1}^n 1/i \approx \frac{6}{\pi^2} \ln n$. Since $\sqrt{\frac{A_1}{A_2}} \sqrt{|\Delta|/3} \geq (|\Delta|/3)^{1/4}$ algorithm 3 traverses about half of the extended Euclidean algorithm applied to A_1/d and R with $A_1, R \leq \sqrt{|\Delta|/3}$. By theorems 1, 2 the number of the remaining steps to reduce (A'_3, B'_3, C'_3) is bounded by a fixed constant. \square

Remarks The standard version of the extended Euclidean algorithm, see Knuth (1980), 4.5.2, applied to $x, y \leq n$ on the average takes about $0.83 \ln n$ iterations. An iteration corresponds to step 2 of algorithm 3. This yields a total of $O(n^2)$ bit operations for the extended Euclidean algorithm on integers $x, y \leq 2^n$, see Knuth (1981), 4.5.2, exercise 30. There exist asymptotically faster gcd-algorithms based on the fast Fourier transform. The algorithm of Schönhage (1971) only takes $O(n (\ln n)^2 \ln \ln n)$ bit operations for n -bit integers.

5. Proof of Theorem 1

The i and the A_3^i in steps 3, 4 of algorithm 3 according to step 4 satisfy

$$A_3^i = (v_1^i)^2 A_3 + v_1^i v_2^i B_3 + (v_2^i)^2 C_3.$$

We conclude from $B_3 = B_2 + 2 A_2 R/d$, $\Delta = B_3^2 - 4 A_3 C_3$, $A_3 = A_1 A_2 / d^2$:

$$\begin{aligned} A_3^i &= (v_1^i)^2 A_1 A_2 / d^2 + v_1^i v_2^i (B_2 + 2 A_2 R/d) + (v_2^i)^2 (B_3 - \Delta) / (4 A_3) \\ &= (v_1^i)^2 A_1 A_2 / d^2 + v_1^i v_2^i (B_2 + 2 A_2 R/d) + (v_1^i)^2 d^2 (4 A_1 A_2)^{-1} [(B_2 + 2 A_2 R/d)^2 - \Delta] \\ &= (v_1^i)^2 A_1 A_2 / d^2 + v_1^i v_2^i (B_2 + 2 A_2 R/d) \\ &\quad + (v_2^i)^2 d^2 (A_1 A_2)^{-1} (A_2 C_2 + B_2 A_2 R/d + A_2^2 R^2 / d^2) \\ &= \frac{d}{A_1} \left[\frac{A_2}{d} \left(\frac{A_1}{d} v_1^i + R v_2^i \right)^2 + v_2^i B_2 \left(\frac{A_1}{d} v_1^i + R v_2^i \right) + (v_2^i)^2 C_2 d \right]. \end{aligned}$$

Induction on i yields $\frac{A_1}{d} v_1^i + R v_2^i = v_3^i$. It follows

$$A_3^i = \frac{d}{A_1} \left[\frac{A_2}{d} (v_3^i)^2 + B_2 v_2^i v_3^i + (v_2^i)^2 C_2 d \right] \quad (1)$$

Since the v_3^i decrease with i , the stop-rule of step 2 implies

$$v_3^i \leq \sqrt{\frac{A_1}{A_2} \sqrt{|\Delta|/3}} \leq v_3^{i-1} = u_3^i \quad (2)$$

Lemma 4 $|u_3^i v_2^i| \leq A_1/d$.

Proof of lemma 4 We first prove by induction on i

$$v_2^i v_2^{i+1} = v_2^i u_2^i \leq 0 \quad \text{for } i \geq 0.$$

The assertion holds for $i=0$ since $u_2^0 = 0$.

Suppose $v_2^{i-1} v_2^i \leq 0$. Step 2 yields for $q^i = \lfloor u_3^i / v_3^i \rfloor \geq 0$:

$$v_2^i v_2^{i+1} = v_2^i (v_2^{i-1} - q^i v_2^i) = v_2^{i-1} v_2^i - q^i (v_2^i)^2 \leq 0,$$

which completes the induction.

$$\begin{vmatrix} u_2^i & u_3^i \\ v_2^i & v_3^i \end{vmatrix} = \begin{vmatrix} 0 & 1 \\ 1 & q^{i-1} \end{vmatrix} \cdots \begin{vmatrix} 0 & 1 \\ 1 & q^0 \end{vmatrix} \begin{vmatrix} u_2^0 & u_3^0 \\ v_2^0 & v_3^0 \end{vmatrix} = \pm A_1/d.$$

Hence $|u_2^i v_3^i - u_3^i v_2^i| = A_1/d$. Clearly, $v_3^i, u_3^i > 0$ and by (2), u_2^i, v_2^i have distinct sign. Therefore

$$|u_3^i v_2^i| \leq |u_3^i v_2^i - u_2^i v_3^i| = A_1/d.$$

We continue the proof of theorem 1. Obviously $|v_3^i| \leq |u_3^i|$ for $i \geq 0$.

Therefore lemma 4 implies

$$|v_3^i v_2^i| \leq A_1/d \quad (3)$$

Moreover

$$|v_2^i| \leq A_1/(d |u_3^i|) \quad (\text{by lemma 4})$$

$$\leq \frac{A_1}{d} \sqrt{\frac{A_2}{A_1 \sqrt{|\Delta|/3}}} \quad (\text{by (2)})$$

Hence

$$|v_2^i|^2 \leq \frac{A_1 A_2}{d^2 \sqrt{|\Delta|/3}} \quad (4)$$

We combine the bounds (2), (3), (4) on $(v_3^i)^2$, $v_2^i v_3^i$, and $(v_2^i)^2$ with (1) and obtain

$$\begin{aligned} A_3' &= \frac{d}{A_1} \left[\frac{A_2}{d} (v_3^i)^2 + B_2 v_2^i v_3^i + C_2 d (v_2^i)^2 \right] \\ &< \frac{d}{A_1} \left[\frac{A_2}{d} \frac{A_1 \sqrt{|\Delta|/3}}{A_2} + \frac{|B_2| A_1}{d} + \frac{A_1 A_2 C_2}{d \sqrt{|\Delta|/3}} \right] \\ &= \sqrt{|\Delta|/3} + |B_2| + A_2 C_2 / \sqrt{|\Delta|/3}. \end{aligned}$$

We have $|B_2| < A_2 < \sqrt{|\Delta|/3}$ and $A_2 C_2 < |\Delta/3|$ since (A_2, B_2, C_2) is reduced. This implies

$$A_3' \leq \sqrt{|\Delta/3|}.$$

6. Proof of Theorem 2

The theorem is true in case $|B| \leq C$. So let us assume $|B| > C$.

Case 1 $C \leq \sqrt{|\Delta|/3}$.

Proof by contradiction: suppose $|\bar{B}| > \bar{A}$. Using $C = \bar{C}$ we obtain

$$\frac{\bar{B}^2 + |\Delta|}{4|\bar{B}|} = \frac{\bar{A}}{|\bar{B}|} \cdot \frac{\bar{B}^2 + |\Delta|}{4\bar{A}} = \frac{\bar{A}}{|\bar{B}|} \cdot C < \sqrt{|\Delta|/3}.$$

This implies

$$\frac{|\bar{B}|}{\sqrt{|\Delta|}} + \frac{\sqrt{|\Delta|}}{|\bar{B}|} < \frac{4}{\sqrt{3}} = \sqrt{3} + \frac{1}{\sqrt{3}}.$$

From this equation we conclude $1/\sqrt{3} < |\bar{B}|/\sqrt{|\Delta|} < \sqrt{3}$, hence $|\bar{B}| > \sqrt{|\Delta|/3}$.

Since $|\bar{B}| \leq \bar{C} = C$ this yields $C > \sqrt{|\Delta|/3}$, a contradiction.

Case 2 $C > \sqrt{|\Delta|/3}$.

Consider the form $(\bar{A}, \bar{B}, \bar{C}) = (A - 2|B| + 4C, |B| - 4C, C)$. It is sufficient to show $|\bar{B}| < C$, $|\bar{B}| < \bar{A}$.

Proof of $|\bar{B}| < C$: We have $|B| \leq A = (B^2 + |\Delta|)/(4C)$ and therefore

$4C - |B| \leq |\Delta/B|$. $|B| \leq A = (B^2 + |\Delta|)/(4C)$ also implies

$\sqrt{|\Delta|/3} < C \leq (|B| + |\Delta|/|B|)/4$, hence $\sqrt{|\Delta|}/|B| + |B|/\sqrt{|\Delta|} > 4/\sqrt{3}$. The

last inequality implies either $|B|/\sqrt{|\Delta|} < 1/\sqrt{3}$ or $|B|/\sqrt{|\Delta|} > \sqrt{3}$.

Since $|B| > C > \sqrt{|\Delta|/3}$ we obtain $|B| > \sqrt{3|\Delta|}$.

This implies $4C - |B| \leq |\Delta/B| < \sqrt{|\Delta|/3}$. On the other hand, by the assumption of theorem 2, we have

$$|B| - 4C < (2.5 - 4/\sqrt{3})\sqrt{|\Delta|} < 0.2\sqrt{|\Delta|} < \sqrt{|\Delta|/3}.$$

Hence $|\bar{B}| = ||B| - 4C| \leq \sqrt{|\Delta|/3} \leq C$.

Proof of $|\bar{B}| \leq \bar{A}$: We conclude from $2.5\sqrt{|\Delta|} \geq A \geq \sqrt{3|\Delta|}$:

$$C = (B^2 + |\Delta|)/(4A) \leq |B|/4 + |\Delta|/(4A)$$

$$\leq A/4 + |\Delta|/(4|B|) \leq \sqrt{|\Delta|}(2.5/4 + 1/(4\sqrt{3})) < 0.77\sqrt{|\Delta|}$$

We conclude from $|B| \leq A \leq 2.5\sqrt{|\Delta|}$, $C \geq \sqrt{|\Delta|}/3$:

$$|B| - 4C \leq 0.2\sqrt{|\Delta|} \leq -A/(4 \cdot 0.77\sqrt{|\Delta|}) \leq -\Delta/(4C)$$

$$\leq \frac{(4C - |B|)^2 - \Delta}{4C} = A - 2|B| + 4C$$

On the other hand $|B| \leq A$ implies $4C - |B| \leq 4C + A - 2|B|$. Hence

$$|\bar{B}| = ||B| - 4C| \leq A - 2|B| + 4C = \bar{A}.$$

□

References

- Gauss, C.F. : *Disquisitiones Arithmeticae*. Leipzig 1801.
German translation: *Untersuchungen über höhere Arithmetik*.
Springer, Berlin (1889).
- Knuth, D.E. : *The Art of Computer Programming, Volume 2, Seminumerical Algorithms*. Second Edition.
Addison-Wesley (1981).
- Lenstra, H.W. Jr. : On the calculation of regulators and class numbers of quadratic fields.
in: J.V. Armitage (ed). *Journées Arithmétiques 1980*, Cambridge University Press (1982), 123-150.
- Schnorr, C.P. and Lenstra, H.W. Jr. : A Monte Carlo factoring algorithm with linear storage.
Mathematics of Computation (1984).
- Schönhage, A. : Schnelle Berechnung von Kettenbruchentwicklungen.
Acta Informatica 1 (1971), 152-157.
- Shanks, D. : Class number, a theory of factorization and genera.
Proc. Symp. Pure Math. Amer. Math. Soc. 20 (1971), 415-440.