Polynomial Selection for the Number Field Sieve Integer Factorisation Algorithm

Brian Antony Murphy

A thesis submitted for the degree of Doctor of Philosophy
of The Australian National University

July 1999

Declaration

Except where otherwise indicated, this thesis is my own original work.

Brian Antony Murphy, July 1999.

Acknowledgments

I have been afforded the rare opportunity of working as a student of Richard Brent. Over the last three years, Richard has provided encouragement, guidance and suggestions from which I have learnt a great deal and for which I am extremely grateful. Richard was also considerate enough to take up a chair in Computing Science at Oxford University in 1998. That gave me an excuse to visit him there, about which I will say more later.

I also owe a great deal to Peter Montgomery (CWI, Amsterdam and Microsoft Research, USA). Peter's influence on current research in this field is far more extensive than most people realise. I have had the benefit of many long discussions with Peter, and a great deal of patient instruction from him. Several key sections of this thesis are developed from ideas originating from discussions with Peter.

My research experience has been enriched and broadened through close collaboration with the Computational Number Theory and Data Security group at CWI in Amsterdam. I thank Herman te Riele, the head of the group, for fostering that collaboration and supporting two visits by me to CWI.

I am also grateful to Arjen Lenstra (Citibank, USA) and John Pollard. Both gentlemen perused early versions of some of the work contained herein and provided valuable comments and suggestions. My thesis would contain many errors and inconsistencies but for some close reading by Mike Newman (ANU) and Adam Hall.

Funding is crucial to research, and I express thanks to all those who have funded mine. The CRC for Advanced Computational Systems (ACSys) funded the first year of my PhD and provided professional development and travel support thereafter. The second and third years of my PhD were funded by an Australian Postgraduate Award. My second and third years were also funded partially by the Defence Science and Technology Organisation (DSTO). I am particularly grateful to Richard Taylor at DSTO for his support and input to my research.

I was very fortunate to be able to spend close to six months visiting Richard Brent at Oxford. I thank Richard, the Oxford University Computing Laboratory, and the providers of the Australian Bicentennial Scholarship for supporting that visit. This was one of most valuable experiences of my PhD.

Let me turn to some personal thanks. Several people have been unfortunate enough to share a house with me in Canberra or Oxford over the past three years. To those of you who are still talking to me, thank you for some wonderful times. I draw particular attention to those with excellent squash or culinary skills.

I need to acknowledge my good friend Rachel Thomas in Perth, because she will be cross if I don't.

Friday nights, in Canberra and Oxford, have been a crucial part of my PhD experience. Thank you to those who enjoyed them with me. Unfortunately, late nights in front of the screen have also been a large part of my PhD experience. Thank you to anyone who made them bearable.

Finally, to my family in Perth, my deepest gratitude for years of support.

Abstract

In this thesis we outline new research in integer factorisation with applications to public-key cryptography. In particular, we consider the number field sieve, the newest and fastest known method for factorising integers used in public-key cryptosystems. We improve so-called *polynomial selection* methods for the number field sieve. Polynomial selection has been a major open problem for the number field sieve since its inception. We address the problem by modelling polynomial yield, and giving methods for finding polynomials with good yield. The improvements described here were used to obtain a new factorisation record, the 140 digit RSA modulus RSA-140, and are being used to obtain a further record by factorising a 512 bit RSA modulus RSA-155.

List of Symbols

The following symbols are used in this thesis without further definition.

$\mathbb{Z}$	the (rational) integers
$\mathbb{Z}_n$	the integers modulo n
$\mathbb{Q}$	the rational numbers
$\mathbb{R}$	the real numbers
$\mathbb{C}$	the complex numbers
$\mathbb{Z}[x]$	the set of polynomials in x with integer coefficients
$\mathbb{Q}(\alpha)$	the number field defined by α, with $\alpha \in \mathbb{C}$ satisfying $f(\alpha) = 0$ for some $f \in \mathbb{Z}[x]$ of degree d
$\mathbb{Z}[\alpha]$	the ring of $\mathbb{Z}$-linear combinations of $\{1, \alpha, \ldots, \alpha^{d-1}\}$ with α as above
$\mathcal{O}$	the ring of (algebraic) integers of $\mathbb{Q}(\alpha)$
$\left(\frac{n}{q}\right)$	the Legendre symbol for n mod q
$\mathrm{ord}_p n$	the exponent of the largest power of p dividing n
$\varphi(n)$	Euler's phi function
$\lfloor r \rfloor$	the floor of $r \in \mathbb{R}$

Contents

Chapter 1

Introduction

Throughout life, and in particular throughout this thesis, N is a large integer requiring factorisation.

Some integers N require factorisation simply because they're large and interesting. Some require factorisation because, as well as being large and interesting, they're important cryptographically. We are concerned with factorising integers of the latter kind.

The most commonly used public-key cryptosystem is the RSA system. The security of RSA relies on certain large N being difficult to factorise. The best measure of the level of security offered by RSA is our ability to factorise such N.

Asymptotically and in practice, the fastest algorithm for factorising these integers is the number field sieve. The speed at which the number field sieve factorises N is determined by the supply of *smooth integers* (integers with no large prime factors) of a particular form. Given a certain pair of polynomials $f_1(x), f_2(x) \in \mathbb{Z}[x]$ each irreducible over $\mathbb{Q}$ and of degree d_i for $i = 1, 2$, we use the homogeneous polynomials $F_i(x, y) = y^{d_i} f_i(x/y)$. We search for coprime integer pairs a, b at which both $F_1(a, b)$ and $F_2(a, b)$ are smooth. This search is the rate determining step in factorising N using the number field sieve.

The area in which the number field sieve has had the greatest capacity for improvement is in the selection of these polynomials. "Better" polynomials are ones which produce more smooth values. We call the problem of choosing better number field sieve polynomials the *polynomial selection problem.*

Motivated by both assessing and compromising the security of RSA and similar systems, we consider in this thesis the polynomial selection problem for the number field sieve. The improvements given here were used to set a new record for factorisation of "general" N, by factorising the 140 digit RSA modulus RSA-140. At the time of writing, our improvements are also being used to factorise a 512 bit (155 digit) RSA modulus RSA-155.

In this chapter we introduce the polynomial selection problem. In Section 1.1 we consider the cryptographic context of the problem. In Section 1.2 we introduce the strategy for factorising integers like RSA moduli. In Section 1.3 we outline very briefly the number field sieve, and focus on the polynomial selection problem. We are then in a position in Section 1.4 to outline the contribution of this thesis, and in Section 1.5 to outline the thesis itself.

1.1 Integer Factorisation and Public-Key Cryptography

Public-key cryptography [28] is a crucial aspect of modern communication networks. Its aim is to ensure that communications over networks are secure.

Most public-key cryptosystems rely for their security on certain number-theoretic problems being intractable. By a large margin, the most commonly used form of public-key cryptography is the RSA cryptosystem. RSA, and some other systems like it, rely for their security on the problem of integer factorisation. Most other public-key cryptosystems in use rely for their security on instances of the discrete logarithm problem. Our results also affect some of these systems.

Next we describe the RSA cryptosystem. We also mention some alternative public-key cryptosystems relying on the discrete logarithm problem. Our treatment is very brief. A good survey of public-key cryptosystems, from the perspective of their underlying number theoretic problem, is [79].

1.1.1 The RSA Public-Key Cryptosystem

In the RSA public-key cryptosystem [68] the public/private-key pair is generated from two distinct large primes p, q of approximately the same size (in fact, they are usually the same number of bits). Let $N = pq$. Choose $e \in \mathbb{Z}$ coprime to $\varphi(N) = (p-1)(q-1)$ and, using for example the extended Euclidean algorithm, compute

$$d \equiv e^{-1} \bmod \varphi(N).$$

The public-key is then the pair (e, N) and the private-key is d. Encryption of the message block M occurs by computing

$$C \equiv M^e \bmod N.$$

Decryption occurs by computing

$$C^d \equiv M^{ed} \equiv M \bmod N.$$

Clearly, factorising N suffices to compromise the security of the system. Also, factorising N is equivalent to factorising $\varphi(N)$ (see for example [11]). Whether the security of RSA is equivalent to factorising N is an open problem. For the paranoid, there are versions of RSA whose security is provably "almost" equivalent to factorising N [66], [84], although these methods suffer other disadvantages [73].

For practical purposes, RSA depends for its security on the difficulty of factorising N, the RSA modulus. Hence, the *level* of security provided by the system depends on our ability to factorise RSA moduli. Current recommendations of modulus length of course depend on the level of security desired. The minimum recommended length for financial and government communications requiring a high level of security is 1024 bits (319 digits). However, 512 bit moduli have been and still are commonly used. For example, 512 bits is the default length on certain Internet browsers, and therefore such moduli protect a large portion of electronic commerce conducted over the Internet. Adi Shamir estimates that 512 bit RSA moduli protect approximately 95 % of Internet electronic commerce [70].

1.1.2 Other Public-Key Systems

After integer factorisation, the most common problem on which the security of public-key cryptosystems is based is the *discrete logarithm problem.* Let G be a finite group. Without loss of generality we can assume G is cyclic, with generator g. Given $a \in G$, the discrete logarithm problem in G is to compute x such that

$$g^x = a.$$

Several cryptographic protocols rely on the discrete logarithm problem for their security, for example the Diffie/Hellman key exchange protocol [28], ElGamal [31], and elliptic curve systems [38].

The most desirable groups G for cryptographic purposes are the ones for which

- the group multiplication law can be implemented efficiently, and
- the discrete logarithm problem in G is believed to be difficult.

Two types of group have emerged in practice as satisfying these requirements; the multiplicative group of a finite field and the group of points on an elliptic curve over a finite field. We denote the first type by $GF(q)^*$ for $q = p^n$, p prime, and the second $E(q)$. In both cases, q will usually be either p for odd p, or 2^n with $n \geq 1$.

There is an analogous version of the number field sieve for factorisation which computes discrete logarithms in some groups of the first type, $GF(q)^*$. Our improvements to the number field sieve for factorisation carry over to this version, and so have an impact on the security of systems relying on these instances of the discrete logarithm problem. We discuss this after giving more details on the number field sieve.

It is significant from the point of view of the security of elliptic curve cryptosystems that there is no known analogue of the number field sieve addressing the elliptic curve discrete logarithm problem. Hence, our improvements do not apply there. Since progress on computing discrete logarithms in $E(q)$ lags behind that on factorisation and discrete logarithms in $GF(q)^*$, elliptic curve cryptography is emerging as the best alternative to RSA.

1.2 Factorisation of General Integers

We focus now on factorisation methods relevant to cryptographic applications.

Not all integers of a given size are equally difficult to factorise. Some are trivial, and some are of a form that makes them susceptible to attack by special methods. In particular, the elliptic curve method [50], [10], and the Pollard-rho method [62] find "small" factors particularly well. The Pollard $p-1$ method [61] succeeds in finding factors p for which $p-1$ contains no large prime factors. For security, RSA moduli need to be integers which are amongst the most difficult to factorise. That is, they need to be integers not susceptible to attack because of their particular form. We refer to integers with no helpful special form as *general integers*. RSA moduli, integers

$$N = pq$$

for primes p and q which are both "large" and not far from $\sqrt{N}$, lie amongst the general integers.

A family of algorithms has been developed for factorisation of general integers. The family is characterised by the factorisation strategy adopted by its members. The number field sieve is the newest and best performing member of the family. Its immediate predecessor is an algorithm called the multiple polynomial quadratic sieve (MPQS) [65], [74]. Several impressive factorisations of RSA moduli were performed using MPQS before the number field sieve came into being. By way of background to the number field sieve we now explain the factorisation strategy of algorithms in the family, concentrating on MPQS and the number field sieve. For a more thorough background on factorisation algorithms the reader should refer to, for example, [67].

1.2.1 Factorisation by Congruent Squares

Long before the introduction of the RSA cryptosystem, Fermat posed the forerunner to the modern strategy. He noted that if positive integers x and y can be found for which

$$N = x^2 - y^2$$

then a non-trivial factorisation of N follows immediately as the product $(x-y)(x+y)$. The task then is to find x and y, that is, to find *the* representation of N as a difference of two squares. The definite article is used advisedly, every odd N which is a product of two prime factors has a *unique* representation as a difference of two squares. Therein lies a problem however; if N is large then finding the unique pair (x, y) becomes difficult.

An improvement usually attributed to Kraithcik [43] partly overcomes this problem. His suggestion is this: instead of requiring N to be a difference of two squares, we should require only *some multiple of* N to be a difference of two squares. There are many multiples of N, so there should be many representations for which to search. Hence, we now require integers x and y such that

$$x^2 \equiv y^2 \bmod N. \tag{1.1}$$

That is, we require two congruent squares mod N. The trade-off is that we are no longer guaranteed that the representation will produce a non-trivial factorisation of N. We *are* guaranteed that $N|(x-y)(x+y)$, from which we may hope that

$$1 < \gcd(x \pm y) < N. \tag{1.2}$$

If for example, $N = pq$ and both p and q divide $x+y$ then (1.2) will not hold. However it is simple to show that amongst a sample of representations (1.1) we can expect (1.2) to hold in at least one half of the cases. Hence we say that given a representation as in (1.1), we get a non-trivial factor of N as in (1.2), with probability at least one half. If N is an RSA modulus then this probability is exactly one half. In practice, finding many representations (1.1) requires only trivially more effort than finding just one, so this does not present an obstacle.

Despite the fact that it is simple, finding two congruent squares mod N becomes the strategy for factorising large general integers. Both MPQS and the number field sieve adopt this strategy. The question now becomes how to construct the congruent squares.

1.2.2 Congruent Squares from Smooth Polynomial Values

Congruent squares are constructed from so-called B-smooth integers.

Definition 1.2.1 *An integer is B-smooth if its largest prime factor is at most B.*

If the precise value of B is immaterial we refer to these simply as smooth integers.

There is a well known procedure for constructing squares from many smooth integers. Let $\pi(B)$ denote the number of primes less than B. Suppose we collect $K > \pi(B)$ integers which are B-smooth. By some linear algebra modulo 2, we can be guaranteed to find a subset of these K integers the product over which is a square.

How does this work? Let the smooth integers we collect be v_i for $i = 1, \ldots, K$. We record the factorisation of each v_i over the primes at most B in an *exponent vector* $\mathbf{v}_i$ of length $\pi(B)$. The j-th entry of $\mathbf{v}_i$ is 1 when the j-th prime appears to an odd exponent in v_i, and 0 otherwise. That is, $\mathbf{v}_i$ records the square-free portion of v_i, with ones denoting the "square-free primes" in v_i. Now we find a subset $\mathcal{S}$ of the v_i's which "pairs-up" the square-free primes appearing throughout the primes $v_i \in \mathcal{S}$. How? We form a matrix over $\mathbb{Z}_2$ listing the $\mathbf{v}_i$ for $i = 1, \ldots, K$ as rows. Since $K > \pi(B)$, the number of columns, there is guaranteed to be a linear dependency, modulo 2, amongst the rows. The set $\mathcal{S}$ now consists precisely of the elements v_i corresponding to the rows $\mathbf{v}_i$ which contribute to the dependency. The product over $\mathcal{S}$ is then guaranteed to be a square, since all the square-free primes are "paired-up" across the product. Hence, from sufficiently many smooth integers, we can construct a square integer after some linear algebra modulo 2. In practice, N is very large and therefore so is the corresponding matrix (typically there are millions of rows and columns).

Recall now that, more than just constructing squares, we are required to construct *congruent squares modulo* N. To ensure that this will be the case on collection of smooth integers, we require the smooth integers to be of a special form. In particular, we require them to be values taken by certain polynomials. For the number field sieve the mechanism by which we are guaranteed to get congruent squares from smooth polynomial values is explained in Chapter 2. For now it suffices to know that both MPQS and the number field sieve proceed by constructing congruent squares modulo N, and that this occurs by collection of sufficiently many smooth values of certain polynomials.

In MPQS, we collect smooth values of certain quadratic polynomials. The number field sieve is more complicated, because there we collect smooth values of *pairs* of polynomials. For example, often one polynomial will be quintic whilst the other is linear. In both MPQS and the number field sieve, the collection of these smooth values is overwhelmingly the most time consuming stage of the process. This stage is called sieving. Other stages are complicated and time consuming too - for example a large matrix requires elimination modulo 2 - but sieving dominates the run-time of the algorithms. The polynomial selection problem for the number field sieve involves reducing the sieving effort required by choosing polynomials which produce many smooth values.

1.2.3 Complexity Estimates

We now consider the relationship between the collection of smooth polynomial values and the run-time of the algorithms. The asymptotic complexity analyses of both MPQS and the number field sieve are tied to the appearance of smooth integers in the context of each algorithm. The results of these analyses give an intuitive picture of where the asymptotic advantage of the number field sieve lies, and what should be exploited to best leverage this advantage in practice.

The following function is central to the analysis. Suppose we have real variables v, w with $0 \leq v \leq 1$. Let the *L-function* be given by

$$L_x[v, w] = \exp\left[\, (w + o(1))\left(\log x)^v (\log\log x)^{1-v}\right)\,\right].$$

The more important variable is v. Think of the L-function as interpolating (along v) between polynomial and exponential functions of $\log x$. Indeed, $L_x[1, w] = x^{w+o(1)}$ and $L_x[0, w] = (\log x)^{w+o(1)}$. The value of w is not immaterial, but makes a difference asymptotically only if v is constant.

N	Date	Algorithm	Effort (MY)	Ref.
RSA-100	Apr 91	MPQS	7	N/A
RSA-110	Apr 92	MPQS	75	[29]
RSA-120	June 93	MPQS	835	[25]
RSA-129	Apr 94	MPQS	5000	[2]
RSA-130	Apr 96	NFS	1000	[23]
RSA-140	Feb 99	NFS	2000	[16]
RSA-155	Sept 99 ?	NFS	< 10000	

Table 1.1: RSA Challenge records

Algorithms in the "congruent squares" family typically have heuristic asymptotic run-times described by the L-function. Heuristically, the time taken by MPQS to factor N is $L_N[1/2, 1]$ as $N \to \infty$. In fact, all general factorisation algorithms preceding MPQS also have asymptotic run-time at best $L_N[1/2, c]$ for some $c \geq 1$. The number field sieve however, has asymptotic run-time

$$L_N[1/3, (64/9)^{1/3}] \; .$$

The appearance of $v = 1/3$ is exciting. It brings the number field sieve significantly closer to a polynomial-time algorithm than its predecessors. We explore this issue in Chapter 2. It is worth noting now that the reason the number field sieve defeats other algorithms asymptotically is that the integers it requires to be smooth are much smaller asymptotically than those of other algorithms. That is, asymptotically, the number field sieve guarantees a much better supply of smooth polynomial values.

1.2.4 The RSA Factorisation Challenge

The number field sieve is clearly the state-of-the-art asymptotically, but what is the situation in practice? State-of-the-art for general integer factorisation in practice is measured by progress through the RSA Factorisation Challenge. The RSA Factorisation Challenge is a list of genuine RSA moduli. The Challenge is administered by RSA Laboratories [69] precisely to encourage and keep track of factorisation research.

The challenge numbers begin at length 100 digits, and there is one at every length $110, 120, \ldots, 500$ digits. There are also moduli of length 155 digits (512 bits), 232 digits (768 bits), 309 digits (1024 bits) and 617 digits (2048 bits).

Table 1.1 shows progress through the list, including the record RSA-140 factorisation and the impending RSA-155 record.

The MIPS-years figures for the effort required to factorise each number are very approximate. For RSA-130 and RSA-140 the figures are a little misleading. Lower, but still conservative, "could-have-done-it-in" estimates for each are 500 and 1500 MIPS-years respectively. The prediction for RSA-155 is based on extrapolation from the 2000 MIPS-years figure for RSA-140. We re-visit this estimate in Chapter 6.

The integers of most interest to us are RSA-129, RSA-130, RSA-140 and RSA-155. RSA-129 is not formally part of the RSA Challenge list, it is included here for illustration purposes only. It was set as a challenge in the August 1977 edition of *Scientific American*. Accompanying the challenge was the now infamous claim that the RSA-129 modulus would be secure for 40 quadrillion years. The RSA-129 factorisation is the last record set using MPQS. The RSA-130 record is significant because it is the first set using the number field sieve. Notice the decrease in estimated MIPS-years effort from RSA-129 to RSA-130. Notice also that the RSA-140 record, even with the conservative estimate of effort, was still set with less than half the effort used for RSA-129.

These figures refer only to the effort spent on the sieving stage of each algorithm; the stage during which smooth polynomial values are collected. Certainly this is the stage that requires the most effort, but particularly with the number field sieve, other stages are also complicated and time consuming.

1.3 The Number Field Sieve Briefly

In this section we outline very briefly the steps involved in the number field sieve. More details are given in Chapter 2. We also note the existence of an analogue of the number field sieve for computing discrete logarithms in $\mathbb{Z}_{p-1}$. Finally we focus on the polynomial selection problem.

The number field sieve was developed from ideas of Pollard [63] in 1988. It was initially formulated to apply to integers of a special form (for which the polynomial selection step is easy). This earlier version is now referred to as the *special number field sieve*. The special number field sieve had early success with the factorisation of the ninth Fermat number F_9 [47]. Since the polynomial selection step is easy in the special number field sieve, that is, an obvious pair of exceptionally good polynomials is known in advance for N, its asymptotic run-time is only

$$L_N[1/3, (32/9)^{1/3}] \ .$$

For completeness we note that the largest integer factorised by the special number field sieve is $10^{211} - 1$ [17].

Our focus is on the polynomial selection step for N where no "special form" polynomials are available. So we do not consider the special number field sieve any further than to say it was extended to apply to general N in [14]. Implementations for general N emerged soon thereafter ([5], [33], [12]). The RSA-140 and RSA-155 factorisations use the implementation of [33], and a variation of [5].

1.3.1 The Algorithm

Suppose we have two polynomials $f_1, f_2 \in \mathbb{Z}[x]$ which are irreducible over $\mathbb{Q}$ and have a common root m mod N. Given $\alpha_i \in \mathbb{C}$ for which $f_i(\alpha_i) = 0$ for $i = 1, 2$, distinct squares are constructed in the number fields $\mathbb{Q}(\alpha_i)$. Viewed in $\mathbb{Z}_N$ as images under homomorphisms defined by sending each $\alpha_i \mapsto m$, these squares give rise to (1.1).

The squares in $\mathbb{Q}(\alpha_i)$ are constructed from smooth values of the homogeneous polynomials $F_i(x, y) = y^{d_i} f_i(x/y)$ where $d_i = \deg f_i$. In fact, coprime integer pairs (a, b) at which both $F_1(a, b)$ *and* $F_2(a, b)$ are smooth are sought. Such a pair is called a *relation*. Relations are identified using a sieving process. Many millions of relations are required for interestingly large values of N. Thus, collecting relations (sieving) is a very time consuming process.

Once sufficiently many relations are collected, a large matrix is constructed over $\mathbb{Z}_2$ much as in the procedure outlined at Section 1.2.2. Finding the squares in each $\mathbb{Q}(\alpha_i)$ requires finding a linear dependency over $\mathbb{Z}_2$ amongst the rows of this matrix.

It is not the squares in $\mathbb{Q}(\alpha_i)$ that are required for (1.1) however, but the homomorphic images in $\mathbb{Z}_N$ of their square roots. To find these images, we require the square root of each square in $\mathbb{Q}(\alpha_i)$. Upon finding those square roots, computing the relevant gcd will, with probability at least one half, produce a non-trivial factor of N.

Thus the number field sieve has the following steps:

1. Polynomial Selection

2. Sieving

3. Matrix Reduction

4. Square Root.

Steps 2–4 are well studied (which is not to suggest that there is no room for improvement). Sieving methods are well developed. That is, we have efficient methods for *detecting* whatever smooth polynomial values are there. The square root step is essentially solved. There exists an algorithm for performing the matrix step, although very large matrices are becoming a problem from an implementation perspective.

The polynomial selection step however, has been a major open problem since the inception of the number field sieve. The problem is to choose polynomials which ensure a good supply of smooth values *in practice*. The main aim in doing so is to decrease sieving times. A pleasant side effect is that we can also, in the presence of other techniques (see Chapter 2), reduce the expected matrix size. Before expanding on polynomial selection in Section 1.3.3, we note another area affected by improvements in polynomial selection.

1.3.2 The Number Field Sieve for Discrete Logarithms

Algorithms for computing discrete logarithms in G fall into two categories; those which work for arbitrary G, and those which rely on properties of particular group representations. We do not discuss the former category here, other than to say that the best known algorithms in this category have run-times that are exponential in $\log h$, where $h = |G|$.

All known sub-exponential algorithms fall into the latter category. They are collectively called *index calculus algorithms*. A survey of index calculus algorithms for discrete logarithm computations is found in [72]. The general strategy of index calculus algorithms is to compute the discrete logarithms of many small elements in G, then express the desired logarithm as a linear combination of the small ones. The first stage, computation of the logarithms of many small elements, is done in a similar manner to general factorisation algorithms. That is, collection of sufficiently many "smooth" elements followed by reduction of a large matrix. (One difference is that the matrix reduction is done modulo l, for each large divisor l of $h - 1$, rather than modulo 2).

For the strategy to work in the discrete logarithm case G must have a representation which admits a notion of smoothness. Examples relevant to cryptography are $GF(p)^* \cong \mathbb{Z}_{p-1}$ for odd p and $GF(q)^*$ where $q = 2^n$ with n large (say, $n \geq 160$). In the former case the representation is simply $\mathbb{Z}_{p-1}$ so smoothness is defined as for integers. In the latter case the representation is the polynomial ring $\mathbb{Z}_2[x]/\langle g(x) \rangle$ for some irreducible polynomial $g \in \mathbb{Z}_2[x]$ of degree n. A polynomial in the ring is considered smooth if its irreducible factors all have small degree compared to n. There is no known sub-exponential attack on elliptic curve discrete logarithms precisely because there is no known analogue of "smoothness" for elements in those groups.

The number field sieve applies to the computation of discrete logarithms in $GF(p)^*$, equivalently, in $\mathbb{Z}_{p-1}$. Prior to the emergence of the number field sieve, the best known algorithm for discrete logarithms in $\mathbb{Z}_{p-1}$ was the Gaussian Integers method of [21]. This method has sub-exponential run-time $L_p[1/2, 1]$. Historically, this method inspired the number field sieve for factorisation. Things turned in a complete circle when Gordon showed in [36] that the number field sieve for factorisation could be applied to compute discrete logarithms in $\mathbb{Z}_{p-1}$. Schirokauer's improvement [71] gives the algorithm now referred to as the number field sieve for discrete logarithms in $\mathbb{Z}_{p-1}$, with heuristic asymptotic run-time

$$L_p[1/3, (64/9)^{1/3}] \ .$$

The most time consuming stage in the discrete logarithm number field sieve is, as with factorisation, the collection of smooth polynomial values. The polynomial selection problem for discrete logarithms is similar to that for factorisation. Any improvements for factorisation therefore, should carry over directly to discrete logarithms.

Record discrete logarithm calculations in $\mathbb{Z}_{p-1}$ lag somewhat behind the corresponding factorisation record. Weber [80] has implemented Schirokauer's algorithm. Using this implementation the record *general* discrete logarithm computation using the number field sieve is that for an 85 digit p in [81]. A larger record, a 129 digit p, was set using the special number field sieve in [82]. The main reason for the lag behind factorisation is that the matrix reduction is more difficult for the discrete logarithm case than for the factorisation case. As mentioned above, the matrix must be reduced modulo some large prime l, not just mod 2.

Our polynomial selection improvements are yet to be applied to the number field sieve for discrete logarithms in $\mathbb{Z}_{p-1}$. We would expect significant improvements once this is done. Not only will sieving time be greatly reduced, but as noted above, better polynomial selection can reduce the expected matrix size.

1.3.3 The Polynomial Selection Problem

So far we have introduced the following:

- The asymptotic advantage of the number field sieve is that its polynomials guarantee a better supply of smooth values than is the case for previous algorithms.
- The polynomial selection problem concerns how to exploit this advantage in practice. The aim is to choose polynomials which generate many smooth values and so reduce the effort required in the time consuming sieving step.
- For N as large as the values we consider in Chapter 6, the matrix step is also troublesome. An advantage of better polynomial selection is that the saving in sieving time is sufficient that, in effect, sub-optimal smoothness bounds can be chosen to decrease the matrix size.

There are essentially two known methods for generating suitable polynomial pairs. For integers as large as, say, RSA-140, a modified base-m method is the better one. With this method, we fix a degree d (for us usually $d = 5$) then seek $m \approx N^{1/(d+1)}$ and a polynomial f_1 of degree d for which

$$f_1(m) \equiv 0 \pmod{N}. \tag{1.3}$$

The polynomial f_1 descends from the base-m representation of N. Indeed, we begin with $f_1(x) = \sum_{i=0}^{d} a_i x^i$ where the a_i are the coefficients of the base-m representation, adjusted so that $-m/2 \leq a_i < m/2$.

The alternative polynomial selection method produces two quadratic polynomials f_1 and f_2. As a function of the degree of the individual polynomials, this method defeats the base-m method. However for sufficiently large integers (like RSA-140) the combined degree of two quadratics f_1 and f_2 is too low to compete with quintic polynomials chosen by the base-m method.

In this thesis we examine polynomials produced by both methods. The problem demands that we choose "good" polynomials. A polynomial's "goodness" is determined by its *yield*, that is, the number of smooth values it produces for a given smoothness bound and in a given range. We consider the problem in three stages; first we decide what to look for, then we decide how to look for it, then we find it.

1.4 Contribution of the Thesis

We characterize the contribution of this thesis into three areas.

1.4.1 Polynomial Yield

Here we decide what to look for. That is, we develop an understanding of polynomial yield. Consider a single polynomial F. We take the yield of F to be influenced by two factors, which we call size and root properties. Choosing good F requires choosing F with a good *combination* of size and root properties.

By *size* we refer to the magnitude of the values taken by F. It has always been well understood that size affects the yield of F.

Definition 1.4.1 *A* random value i_r *is an integer chosen uniformly at random from* $\{i \in \mathbb{Z} : 1 \leq i \leq r\}$.

For a fixed smoothness bound the likelihood of a random value i_r being smooth decreases rapidly as $r \to \infty$, and does so in a well known manner. Hence, previous approaches to polynomial selection have sought polynomials whose size is smallest.

The influence of root properties however has not been either well understood or adequately exploited. By *root properties* we refer to the distribution of the roots of F modulo small p^k for p prime and $k \geq 1$. In short, if F has many roots modulo small p^k, values of F "behave" as if they are smaller than they actually are.

We contribute an understanding of this effect by quantifying root properties and modelling the interaction between size and root properties to determine polynomial yield. Only once this is done can we know what is required for a particular polynomial to have "good" yield.

1.4.2 Polynomial Selection

Once we understand what to look for, we develop methods for finding it. We seek methods for generating polynomials with good combinations of size and root properties. We contribute some tricks and techniques which help find such polynomials. As part of this process we also contribute techniques for determining, *without* sieving, the "goodness" of a particular polynomial.

1.4.3 Polynomials for Record Factorisations

We used the improvements given in this thesis to select polynomials for the record factorisation of RSA-140. We found a decrease by a factor of two in the expected sieving time (extrapolated from RSA-130), because of the improved selection. We found a decrease in the expected matrix size (that is, the number of rows or columns) by a factor of about 1.4, because of the improved selection in the context of other procedures (see Chapter 2). We used a polynomial pair whose yield is approximately 8 times that of a random selection.

We made further and better use of our techniques for the factorisation of RSA-155. At the time of writing, the sieving task for RSA-155 is complete. We used a polynomial pair whose yield is approximately 13.5 times that of a random selection. We expect the factorisation in August/September of 1999.

Factorisation of a 512 bit RSA modulus is a significant milestone in integer factorisation for cryptographic purposes, and effectively renders such moduli useless for serious applications.

1.5 Outline of the Thesis

Chapter 2 contains mainly background material. In discussing the background material we survey the relevant literature. The focus is on aspects most relevant to polynomial selection.

Chapters 3–6 contain the bulk of the research. Chapters 3 and 4 are aimed at developing our understanding of polynomial yield. Chapter 3 establishes the framework mainly by parameterising root properties. Chapter 4 considers the effect of size and root properties together. Initially we examine yield as a function of root properties to check their effect and our paramaterisation. We then give a simple method of estimating yield. Material contained in Chapters 3 and 4 appeared in [56] and [57].

In Chapters 5–6 we use our understanding of yield to address the polynomial selection problem. Chapter 5 contains techniques for generating good polynomials. The focus is on polynomials relevant to factorisation of large RSA moduli. In Chapter 6 we investigate these techniques, by re-examining the factorisation of RSA-130 and by describing the polynomial selection for the RSA-140 and RSA-155 factorisations. Some of the material contained in Chapters 5 and 6 appears in [16] and some was delivered in [54].

Chapter 7 contains conclusions and suggestions for further work.

Chapter 2

Background

In this chapter we give background material concerning the number field sieve. In doing so, we survey the relevant literature. The focus is on issues directly relevant to the polynomial selection problem.

In Sections 2.1 and 2.2 we examine the number field sieve broadly. In Section 2.1 we focus on the algorithm, its algebraic context, and its practical stages. In Section 2.2 we examine the asymptotic complexity analysis of the algorithm. In Section 2.3 we consider the polynomial selection problem specifically.

2.1 The Number Field Sieve

We saw a very brief description of the number field sieve in Chapter 1. Here we elaborate, concentrating on matters relevant to polynomial selection.

In Section 2.1.1 we give a more detailed overview of the algorithm than that in Chapter 1. We see from this overview that the algorithm lies in a complicated algebraic context. The relevance of the smooth polynomial values is clear once placed in this context. In Section 2.1.2 we survey the results on this issue. Since we then understand why smooth polynomial values are important, we turn in Section 2.1.3 to how they are found. That is, we discuss sieving methods. In Section 2.1.4 we consider the matrix step. Rather than describe the algorithms used for reduction of the matrix, we focus on the benefit received at the matrix stage from better polynomial selection. We illustrate with the factorisation of RSA-140. Finally, in Section 2.1.5 we point to the literature concerning the square-root stage.

2.1.1 Outline

Let $f_1(x)$ and $f_2(x) \in \mathbb{Z}[x]$ be irreducible (over $\mathbb{Z}$) polynomials. For ease of exposition, for the moment we assume f_1 and f_2 are monic. We will relax this assumption soon. Also, suppose there exists some $m \in \mathbb{Z}$ for which

$$f_1(m) \equiv f_2(m) \equiv 0 \bmod N.$$

That is, f_1 and f_2 have a common root m mod N. This requirement is very limiting. Finally, suppose that α_1, α_2 are complex roots of f_1 and f_2 respectively, with corresponding number fields $\mathbb{Q}(\alpha_1)$ and $\mathbb{Q}(\alpha_2)$. Think of m as the analogue mod N of each $\alpha_i \in \mathbb{C}$ for f_i. The key point is that both f_1 and f_2 have the same analogue.

We define ring homomorphisms $\varphi_1 : \mathbb{Z}[\alpha_1] \to \mathbb{Z}_N$ and $\varphi_2 : \mathbb{Z}[\alpha_2] \to \mathbb{Z}_N$ by sending $\alpha_1 \mapsto m$ mod N and $\alpha_2 \mapsto m$ mod N. For example

$$\varphi_1\left(\sum_{i=0}^{d-1} a_i \alpha_1^i\right) = \sum_{i=0}^{d-1} a_i m^i \bmod N$$

with $a_i \in \mathbb{Z}$ being the coefficients of f_1 and d the degree of f_1.

Suppose that there exists a set $\mathcal{S}$ of coprime integer pairs (a, b) for which both

$$\begin{aligned}\prod_{(a,b)\in\mathcal{S}} (a - b\alpha_1) &= \beta_1^2 \text{ for some } \beta_1 \in \mathbb{Z}[\alpha_1], \text{ and}\\ \prod_{(a,b)\in\mathcal{S}} (a - b\alpha_2) &= \beta_2^2 \text{ for some } \beta_2 \in \mathbb{Z}[\alpha_2].\end{aligned}$$

Then

$$\begin{aligned}\varphi_1(\beta_1^2) &= \prod_{(a,b)\in\mathcal{S}} (\varphi_1(a - b\alpha_1)) \equiv \prod_{(a,b)\in\mathcal{S}} (a - bm) \bmod N, \text{ and}\\ \varphi_2(\beta_2^2) &= \prod_{(a,b)\in\mathcal{S}} (\varphi_2(a - b\alpha_2)) \equiv \prod_{(a,b)\in\mathcal{S}} (a - bm) \bmod N,\end{aligned}$$

giving

$$\varphi_1(\beta_1)^2 \equiv \varphi_2(\beta_2)^2 \bmod N.$$

The key point is that starting with polynomials which have a common root mod N ensures that the squares $\varphi_1(\beta_1)^2$ and $\varphi_2(\beta_2)^2$ are congruent mod N. Now $\gcd(\varphi_1(\beta_1) \pm \varphi_2(\beta_2), N)$ will be a non-trivial factor of N with probability at least $1/2$, in the sense described in Section 1.2.1.

The question therefore becomes how to construct the set $\mathcal{S}$. It is here that smooth polynomial values become relevant. Associated with each polynomial f_i is the binary homogeneous polynomial

$$F_i(x, y) = y^d f_i(x/y). \tag{2.1}$$

Where possible, which is not often, we omit the subscript i.

Note 2.1.1 Throughout this thesis, we use the upper case F to denote the homogenous binary version given in (2.1) of the corresponding lower case $f \in \mathbb{Z}[x]$.

Our set $\mathcal{S}$ is constructed by collecting smooth values of the polynomials F_i. In particular, we collect coprime integer pairs (a, b) at which both $F_1(a, b)$ and $F_2(a, b)$ are B-smooth for some smoothness bound B. We call such an (a, b) pair a *relation*. In fact it suffices if either of $F_1(a, b)$ or $F_2(a, b)$ is *almost* smooth, as we discuss in Section 2.1.3. By a simple extension of the linear algebra procedure we saw in Section 1.2.2, from enough (a, b) relations, a set $\mathcal{S}$ of (a, b) can be found for which each

$$\prod_{(a,b)\in\mathcal{S}} F_i(a, b) \tag{2.2}$$

is square in $\mathbb{Z}$. Furthermore, by the construction we explain below, each product (2.2) being square in $\mathbb{Z}$ makes it practically certain that each

$$\prod_{(a,b)\in\mathcal{S}} (a - b\alpha_i) \tag{2.3}$$

is the required square $\beta_i^2 \in \mathbb{Z}[\alpha_i]$. The implication from (2.2) to (2.3) is certainly not obvious in advance.

In practice, a sieving process is used to identify relations. This is the so-called *sieving stage*. Since many smooth values are required, and smooth values are rare, this is overwhelmingly the most time consuming stage of the algorithm. Indeed, the time taken by sieving dominates the run-time analysis of the entire algorithm. That is why good polynomial selection is so crucial to decreasing the time taken to factorise N: good polynomial selection increases the number of smooth values produced by F.

On completion of sieving we find $\mathcal{S}$ by reduction modulo 2 of a large sparse matrix. Although this does not require as much CPU effort as sieving, the matrix reduction is a highly non-trivial process.

On construction of $\mathcal{S}$ we have a product of millions of large algebraic numbers (typically a, b are also in the order of millions). The product is a square $\beta_i^2 \in \mathbb{Z}[\alpha_i]$. We need β_i. Hence, a square root of the large algebraic number β_i^2 must be taken. This also is a non-trivial exercise.

Thus, the number field sieve has the following steps.

1. **Polynomial Selection:** Select f_1 and f_2 (equivalently, F_1 and F_2), with a common root mod N to produce many smooth values.

2. **Sieving:** Collect relations. That is, find coprime (a, b) at which both $F_1(a, b)$ and $F_2(a, b)$ are B-smooth, or almost B-smooth, for some bound B.

3. **Matrix Reduction:** Reduce a large sparse matrix over $\mathbb{Z}_2$ to find the required set $\mathcal{S}$.

4. **Square Root:** Given $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha_i) = \beta_i^2$ for some $\beta_i \in \mathbb{Z}[\alpha_i]$, find β_i (and hence $\varphi_i(\beta_i)$).

On completion of Step 4, computing $\gcd(\varphi_1(\beta_1) \pm \varphi_2(\beta_2), N)$ will, with probability at least 1/2, give a non-trivial factor of N. Construction of one set $\mathcal{S}$ corresponds to finding one linear dependency amongst the rows of the large matrix. Finding several dependencies costs only trivially more time, so in fact we find sufficiently many sets $\mathcal{S}$ to make us practically certain of obtaining a non-trivial factor of N at this stage.

It remains to justify the implication from (2.2) to (2.3). We do this in the next subsection. In doing so we also relax some of our simplifying assumptions.

2.1.2 Congruent Squares from Smooth Polynomial Values

We assume some familiarity with algebraic number theory. Useful background references are [75], [8] and [18].

We have the polynomial $f(x)$ (at this stage we still assume that f is monic) with α such that $f(\alpha) = 0$, the number field $\mathbb{Q}(\alpha)$ whose ring of algebraic integers is $\mathcal{O}$, and we consider elements $a - b\alpha \in \mathbb{Q}(\alpha)$. Ultimately we wish to extract enough information (by sieving) about the multiplicative structure in $\mathbb{Q}(\alpha)$ of each $a - b\alpha$, to deduce that the product over some set of $a - b\alpha$ is square.

Multiplicative structure in $\mathcal{O}$ is difficult to visualise. Moreover, sieving over elements in $\mathbb{Q}(\alpha)$ is a complicated proposition. Multiplicative structure in $\mathbb{Z}$ however, is easy to visualise, and sieving over integers is an entirely attractive proposition. We make the transition from $\mathbb{Q}(\alpha)$ to $\mathbb{Z}$ using the norm map. Usually the norm of an element $\zeta \in \mathbb{Q}(\alpha)$ tells *something* about the multiplicative structure in $\mathbb{Q}(\alpha)$ of ζ. It turns out that for $\zeta \in \mathbb{Q}(\alpha)$ of the particular form $\zeta = a - b\alpha$, the norm tells *everything* about the multiplicative structure in $\mathbb{Q}(\alpha)$ of ζ.

That is, complete information about the ideal factorisation of $\langle a - b\alpha \rangle$ (the ideal generated by $a - b\alpha$) can be deduced from the integer factorisation of its norm. Now, it also turns out that the norm of each element $a - b\alpha$ is given by the value of our homogeneous polynomial, $F(a, b)$. Hence, the integer factorisation of $F(a, b)$ (which we discover by sieving) gives information on the multiplicative structure of $\langle a - b\alpha \rangle$, and that information suffices in practice to construct the requisite square in $\mathbb{Q}(\alpha)$.

In the remainder of this subsection we elaborate on this argument, essentially giving an exposition of the results in [14] and elsewhere. After recalling some basic facts, we explain the argument in a simple case. That is, we assume that in $\mathbb{Q}(\alpha)$ we have $\mathbb{Z}[\alpha] = \mathcal{O}$. In general of course possibly $\mathbb{Z}[\alpha] \subset \mathcal{O}$. We refer to number fields in which the assumption holds as *convenient number fields*. We then relax the assumption, and consider the argument in what we call *arbitrary number fields*. That requires the use of a probabilistic device using *quadratic characters*. Finally, we relax the assumption that f is monic.

Convenient Number Fields

Under the assumption that $\mathcal{O} = \mathbb{Z}[\alpha]$ we have available in $\mathbb{Z}[\alpha]$ the full theory of unique factorisation into prime ideals of ideals in $\mathcal{O}$. Recall the following basic facts.

- For an element $\zeta \in \mathbb{Q}(\alpha)$, the *norm* $\mathbf{N}$ of ζ is given by,

$$\mathbf{N}(\zeta) = \prod_{i=1}^{d} \sigma_i(\zeta),$$

  where d is the degree of $\mathbb{Q}(\alpha)$ (and of f) and the σ_i are the d embeddings of $\mathbb{Q}(\alpha)$ in $\mathbb{C}$. The norm $\mathbf{N}$ is multiplicative.

- For an ideal $\mathfrak{q}$ of $\mathcal{O}$, the *norm* $\mathfrak{N}$ of $\mathfrak{q}$ is given by

$$\mathfrak{N}\mathfrak{q} = |\mathcal{O}/\mathfrak{q}| .$$

  The norm $\mathfrak{N}$ is multiplicative. Moreover,

$$\mathfrak{N}\langle\zeta\rangle = \mathbf{N}(\zeta)$$

  where $\langle\zeta\rangle$ denotes the ideal generated by ζ.

- For every non-trivial prime ideal $\mathfrak{p}$ of $\mathcal{O}$, we have $\mathfrak{N}\mathfrak{p} = p^{\delta}$ for a unique (rational) prime p, and for some postive integer δ. We call δ the *degree* of the ideal $\mathfrak{p}$.

- Rational primes p decompose in $\mathcal{O}$ as follows;

$$\langle p\rangle = \prod_{i=1}^{g} \mathfrak{p}_i^{e_i},$$

  for positive integers e_i. Each e_i is called the *ramification index* of p at $\mathfrak{p}_i$. With δ_i being the degree of $\mathfrak{p}_i$ we have

$$\sum_{i=1}^{g} e_i\delta_i = d.$$

  A prime p for which some $e_i > 1$ is called a *ramified* prime.

Let $e_p(\zeta) = \mathrm{ord}_p\mathbf{N}(\zeta)$. We have

$$\prod_{p} p^{e_p(\zeta)} = \mathbf{N}(\zeta) = \mathfrak{N}\langle\zeta\rangle = \mathfrak{N}(\prod_{\mathfrak{p}} \mathfrak{p}^{v_{\mathfrak{p}}(\zeta)})$$

where $\mathfrak{p}$ ranges across the prime ideals of $\mathcal{O}$, p ranges across the rational primes, and for positive integers $v_{\mathfrak{p}}(\zeta)$ (the $\mathfrak{p}$-adic valuations of ζ). We are now interested in the relationship between $e_p(\zeta)$ and $v_{\mathfrak{p}}(\zeta)$ when ζ is square. Since factorisation into prime ideals in $\mathcal{O}$ is unique, ζ is square if and only if every $v_{\mathfrak{p}}(\zeta)$ is even. For this to be the case it is necessary, but not sufficient, that every $e_p(\zeta)$ is even. If either

- the rational prime p is contained in more than one, say two, distinct prime ideals $\mathfrak{p}_i$ and $\mathfrak{p}_j$, or
- $\mathfrak{N}(\mathfrak{p}) = p^\delta$ for some $\delta > 1$,

then $e_p(\zeta)$ can be even whilst $v_{\mathfrak{p}}(\zeta)$ is not.

We avoid the first obstruction by keeping track of appearances of p in $\mathbf{N}(\zeta)$ in a more particular manner than just $e_p(\zeta)$. We will use exponents $e_{p,r}(\zeta)$, with possibly more than one r for each p, and insist that each of these are even in our purported square rather than just each $e_p(\zeta)$ begin even. The second obstruction is avoided by the special form of our elements $\zeta = a - b\alpha$.

The following theorem gives the means for overcoming the first obstruction. It can be found in for example [14] and [48], or viewed as a consequence of Theorem 4.8.13 of [18].

Theorem 2.1.2 *Let p be a rational prime and let*

$$\mathcal{R}(p) = \{r \in \mathbb{Z}_p : f(r) \equiv 0 \bmod p\}.$$

Then the first degree prime ideals of $\mathcal{O}$ with norm p are in one-to-one correspondence with the pairs (p, r) for $r \in \mathcal{R}(p)$.

Since we are assuming that $\mathbb{Z}[\alpha] = \mathcal{O}$, we can index the first degree primes of $\mathbb{Z}[\alpha]$ by the pairs (p, r). In fact, for $\mathfrak{p} \leftrightarrow (p, r)$ and $\zeta = \sum_{i=0}^{d-1} a_i\alpha^i$ in $\mathbb{Z}[\alpha]$,

$$\mathfrak{p}|\zeta \quad\Leftrightarrow\quad \sum_{i=0}^{d-1} a_i r^i \equiv 0 \bmod p. \tag{2.4}$$

The following theorem, found for example in [14] and [47], overcomes the second obstruction.

Theorem 2.1.3 *For coprime integers a and b, every prime ideal of $\mathcal{O}$ dividing $\langle a - b\alpha\rangle$ is a first degree prime ideal.*

For the particular elements $a - b\alpha$ we obtain from (2.4) and Theorem 2.1.3 that

$$a - b\alpha \in \mathfrak{p} \quad\Leftrightarrow\quad a - br \equiv 0 \bmod p. \tag{2.5}$$

So, we now have an exact correspondence between the valuations of ideals $\mathfrak{p} \leftrightarrow (p, r)$ dividing $a - b\alpha$ and the exponent $e_{p,r}$ of $\mathfrak{p}$ in $\mathbf{N}(a - b\alpha)$. That is,

$$\mathbf{N}(a - b\alpha) = \mathfrak{N}(\prod_{\mathfrak{p}} \mathfrak{p}^{v_{\mathfrak{p}}(a-b\alpha)}) = \prod_p p^{e_{p,r}(a-b\alpha)},$$

with $v_{\mathfrak{p}}(a - b\alpha) = e_{p,r}(a - b\alpha)$. Hence, we have the desired correspondence between norm factorisations and multiplicative structure in $\mathbb{Q}(\alpha)$.

The following theorem ties the norm values to the values of F (see for example [83]).

Theorem 2.1.4 *In $\mathbb{Q}(\alpha)$ we have $F(a,b) = \mathbf{N}(a - b\alpha)$.*

We can now describe the sieving framework for F.

Definition 2.1.5 *An element $\zeta \in \mathbb{Q}(\alpha)$ is B-*smooth *exactly when its norm is B-smooth.*

Smooth elements $a - b\alpha \in \mathbb{Z}[\alpha]$ can be found by sieving over the polynomial values $F(a,b)$ with first degree prime ideals of $\mathbb{Z}[\alpha] = \mathcal{O}$, indexed by (p,r), by checking the right hand side of (2.5). The *factor base* consists of all first degree prime ideals of $\mathbb{Z}[\alpha]$ with norm $p < B$ for some bound B. Given sufficiently many B-smooth $a - b\alpha$, by linear algebra over $\mathbb{Z}_2$ there exists a set $\mathcal{S}$ of (a,b) for which

$$\sum_{(a,b)\in\mathcal{S}} e_{p,r}(a - b\alpha) \equiv 0 \bmod 2 \tag{2.6}$$

for all primes $\mathfrak{p} \leftrightarrow (p,r)$ in the factor base. If $\mathcal{O} = \mathbb{Z}[\alpha]$, then (2.6) is sufficient to guarantee that $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha)$ is a square in $\mathbb{Z}[\alpha]$.

Arbitrary Number Fields

We now drop the pretence that $\mathcal{O} = \mathbb{Z}[\alpha]$. We no longer have at our disposal in $\mathbb{Z}[\alpha]$ the theory of uniqueness of factorisation into prime ideals in $\mathcal{O}$. Instead we look more generally to prime ideals of arbitrary *orders* of $\mathbb{Q}(\alpha)$. Denote $\mathbb{Q}(\alpha)$ by K. An order is a subring of K which as a $\mathbb{Z}$-module is finitely generated and of rank $d = \deg K$ (see [18] Section 4.6). Clearly $\mathbb{Z}[\alpha]$ is an order. The maximal order $\mathcal{O}$ satisfies properties not satisfied by arbitrary orders A. In [14] the authors present results which tie the ideal structure in arbitrary orders A to the known structure in $\mathcal{O}$. We are interested in the case $A = \mathbb{Z}[\alpha]$, with the aim of re-establishing the connection between norm factorisations of $F(a,b)$ and ideal factorisations in A.

The following result from [14] introduces homomorphisms $l_\mathfrak{p}$. Think of the $l_\mathfrak{p}$ as generalisations, to arbitrary orders, of $\mathfrak{p}$-adic valuations in $\mathcal{O}$.

Theorem 2.1.6 *Let A be an order of $\mathcal{O}$. There is, for each prime $\mathfrak{p}$ of A, a group homomorphism $l_\mathfrak{p} : K^* \to \mathbb{Z}$, such that the following hold:*

1. *$l_\mathfrak{p} \geq 0$ for all $\zeta \in A$, $\beta \neq 0$;*
2. *if $\zeta \in A$ and $\zeta \neq 0$, then $l_\mathfrak{p}(\zeta) > 0$ if and only if $\zeta \in \mathfrak{p}$;*
3. *for all $\zeta \in K^*$ we have $l_\mathfrak{p}(\zeta) = 0$ for all but finitely many $\mathfrak{p}$, and*

$$\prod_{\mathfrak{p}} \mathfrak{N}\mathfrak{p}^{l_\mathfrak{p}(\zeta)} = |\mathbf{N}(\zeta)|,$$

where $\mathfrak{p}$ ranges over the primes of A.

The functions $l_{\mathfrak{p}}$ are deduced from the structure in $\mathcal{O}$ as follows. Let $\mathfrak{q}$ be a prime in $\mathcal{O}$ lying above the prime $\mathfrak{p}$ in A (that is, $\mathfrak{p} = A \cap \mathfrak{q}$). The field $\mathcal{O}/\mathfrak{q}$ is an extension of $A/\mathfrak{p}$ of degree δ say. Then, informally, $l_{\mathfrak{p}}$ counts δ appearances of $\mathfrak{p}$ in A for every appearance of $\mathfrak{q}$ in $\mathcal{O}$. Applying Theorem 2.1.6 to $\zeta = a - b\alpha$ gives the following.

Corollary 2.1.7 *Let a and b be coprime integers and let $\mathfrak{p}$ be a prime of $\mathbb{Z}[\alpha]$. If $\mathfrak{p}$ is not a first degree prime then $l_{\mathfrak{p}}(a - b\alpha) = 0$. If $\mathfrak{p}$ is a first degree prime corresponding to the pair (p, r) then $l_{\mathfrak{p}}(a - b\alpha) = e_{p,r}(a - b\alpha)$.*

Hence, we have again captured an exact correspondence between the integer factorisation of $\mathbf{N}(a - b\alpha)$ and the ideal factorisation of $\langle a - b\alpha\rangle$. Now, by linear algebra over $\mathbb{Z}_2$ we are able to find a set $\mathcal{S}$ of coprime pairs (a, b) for which

$$\sum_{(a,b)\in\mathcal{S}} e_{p,r}(a - b\alpha) \equiv 0 \bmod 2. \tag{2.7}$$

We saw previously, by uniqueness of factorisation of prime ideals in $\mathcal{O}$, that if $\mathcal{O} = \mathbb{Z}[\alpha]$ then (2.7) is sufficient to guarantee that $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha)$ is a square in $\mathbb{Z}[\alpha]$. This is not the case in general for the following reasons (from [14]).

1. $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha)\mathcal{O}$ may not even be a square in $\mathcal{O}$, since we have considered only primes of $\mathbb{Z}[\alpha]$.
2. Even if $\prod_{(a,b)\in\mathcal{S}}(a-b\alpha)\mathcal{O}$ *is* a square in $\mathcal{O}$, it may not be the square of a principal ideal in $\mathcal{O}$.
3. Even if $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha)\mathcal{O}$ is the square of a principal ideal in $\mathcal{O}$, $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha)$ is not necessarily a square generator.
4. Even if $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha)$ is a generator β^2 (for some $\beta \in \mathcal{O}$) of a principal square ideal, β does not necessarily lie in $\mathbb{Z}[\alpha]$.

Obstruction 4 can easily be overcome, since if $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha)$ is a square in $\mathcal{O}$ then

$$f'(\alpha)^2 \cdot \prod_{(a,b)\in\mathcal{S}} (a - b\alpha) = \omega^2$$

for some $\omega \in \mathbb{Z}[\alpha]$. The remaining obstructions are overcome using quadratic characters.

Quadratic Characters

The use of quadratic characters was first suggested by Adleman [1]. Let V be the (multiplicative) group of $\zeta \in K^*$ for which $l_{\mathfrak{p}}(\beta) \equiv 0 \bmod 2$ for all primes $\mathfrak{p}$ of $\mathbb{Z}[\alpha]$.

That is, V contains the elements which, judging by the primes of $\mathbb{Z}[\alpha]$, *look* like squares in K. Not all of them are, so $K^{*2} \subset V$ where K^{*2} is the multiplicative group of squares in K. Now, V/K^{*2} forms a vector space over $\mathbb{Z}_2$. In [14] (Theorem 6.7) it is shown that there exists a "small" upper bound on the dimension of the vector space. Since V is in that sense not too much larger than K^{*2}, it is plausible that there exists a probabilistic method by which, given $\zeta \in V$, it is practically certain that in fact the element $\zeta \in K^{*2}$. Quadratic characters give us such a method.

Let q be an odd prime and let $s \in \mathcal{R}(q)$ be such that the ideal (q, s) does not lie in the factor base. Also, let

$$\chi_{\mathfrak{q}}(a - b\alpha) = \left(\frac{a - bs}{q}\right).$$

The essence of using quadratic characters $\chi_{\mathfrak{q}}$ is the following. We can be practically certain that if $\zeta \in \mathbb{Z}[\alpha]$ satisfies $\chi_{\mathfrak{q}}(\zeta) = 1$ for sufficiently many first degree primes $\mathfrak{q}$ not lying above ζ, then ζ is a square in K.

In practice, extra columns are annexed to the matrix over $\mathbb{Z}_2$ whose rows represent the relations. Each extra column corresponds to a test prime q. The entry in the row corresponding to entry (a, b) and column correpsonding to q is 1 if $\chi_{\mathfrak{q}}(a + b\alpha) = -1$ and zero otherwise. Hence a linear dependency amongst the rows ensures

$$\chi_{\mathfrak{q}}\left(\prod_{(a,b)\in\mathcal{S}} (a - b\alpha)\right) = \prod_{(a,b)\in\mathcal{S}} \left(\frac{a - bs}{q}\right) = 1$$

for all test primes q. If sufficiently many q are chosen, $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha)$ is almost certainly a square in $\mathbb{Z}[\alpha]$. Obstructions 1–3 are overcome in one hit, and the relationship in (2.7) captures square appearances of ideals.

Non-monic Polynomials

So far we have assumed f and F are monic. Allowing f to be non-monic gives smaller coefficients - some of the "size" of the coefficients can be pushed onto the leading coefficient. For example, the monic base-m method gives $m = O(N^{1/d})$ and $a_i = O(N^{1/d})$. Non-monic base-m polynomials have $m = O(N^{1/(d+1)})$ and $a_i = O(N^{1/(d+1)})$. It is crucial that we are again able to capture the correspondence between the integer factorisation of $\mathbf{N}(a - b\alpha)$ and the ideal factorisation of $a - b\alpha$. Allowing non-monic polynomials requires some minor adjustments, and we outline these below.

The significance to this point of f being monic is that α being a root of f guarantees that $\alpha \in \mathcal{O}$, so $\mathbb{Z}[\alpha]$ is an order of $\mathcal{O}$. If $a_d \neq \pm 1$ that is not necessarily the case. It turns out however that $A = \mathbb{Z}[\alpha] \cap \mathbb{Z}[\alpha^{-1}]$ *is* an order of $\mathcal{O}$ (see [14]). So Theorem 2.1.6 is again available.

Let ω be a zero of $F(x, a_d)$. If $\alpha = \omega/a_d$ then

$$F(\omega, a_d) = 0 \quad \Rightarrow \quad F(\alpha, 1) = f(\alpha) = 0$$

since F is homogeneous. Now, $\mathbb{Z}[\omega]$ is an order ($\omega \in \mathcal{O}$), and $a_d(a - b\alpha) = a_d a - b\omega$. Also,

$$F(a, b) = \mathfrak{N}(a_d a - b\omega),$$

so we have

$$F(a, b) = a_d \mathfrak{N}(a - b\alpha),$$

compared with Theorem 2.1.4.

Recall that in the monic case we are able to index the first degree primes of $\mathbb{Z}[\alpha]$ by pairs (p, r) for $r \in \mathcal{R}(p)$. If we now identify r with r_1/r_2 whenever $r_2 \neq 0$ then it makes sense to consider the set $\{(r_1, r_2) \in \mathbb{Z}_p^2 : F(r_1, r_2) \equiv 0 \bmod p\}$. In fact we define

$$\mathcal{R}'(p) = \left\{(r_1, r_2) \in \mathbb{Z}_p^2 : F(r_1, r_2) \equiv 0 \bmod p\right\} \cup \{\infty\},$$

and in the case $r_2 = 0$ we identify $r \in \mathcal{R}(p)$ with $\infty \in \mathcal{R}'(p)$. Now, let $e_{p,r}(a, b)$ as before denote the exponent in $\mathbf{N}(a - b\alpha)$ corresponding to the ideal (p, r). Then Theorem 2.1.6 admits homomorphisms $l_\mathfrak{p}$ (where $\mathfrak{p}$ ranges across the prime ideals of A) for which

$$e_{p,r}(a, b) = \begin{cases} l_\mathfrak{p}(a - b\alpha) & \text{if } r \neq \infty \\ l_\mathfrak{p}(a - b\alpha) + \mathrm{ord}_p a_d & \text{if } r = \infty, \end{cases}$$

(see [14] and [59]).

So, with non-monic polynomials, we again capture the correspondence between norm and ideal factorisations that gives rise to a practical method of sieving. Thus, the significance of smooth polynomial values is that they carry enough information on the multiplicative structure in $\mathbb{Q}(\alpha)$ of the corresponding elements, to enable the construction of squares in $\mathbb{Z}[\alpha]$.

Aside 2.1.8 For another application which uses this correspondence, see the literature regarding practical solution of Thue-Mahler equations [76].

Since we now understand how sieving corresponds to deducing multiplicative structure in $\mathbb{Q}(\alpha)$, we should consider how sieving occurs.

2.1.3 The Sieving Step

In this subsection we survey sieving techniques and variations thereof which are relevant to the factorisations discussed in Chapter 6. The main sieving techniques are lattice sieving and line sieving, and the main variations are large prime variations.

Sieving Methods

The use of sieving techniques in factorisation was first proposed for the quadratic sieve. Indeed, this innovation allowed the quadratic sieve to out-perform its competitors at the time. The idea is that given p and a polynomial $W(x)$, the integer values of x for which $p|W(x)$ are regularly spaced mod p. Start with an array of $W(x)$ values for x in some range, and a particular x_0 at which $p|W(x_0)$, the remaining x at which $p|W(x)$ satisfy

$$x \equiv x_0 \bmod p.$$

Moreover, division of $W(x)$ by p can be mimicked by subtraction from $\log W(x)$ of $\log p$. So, starting instead with an array of $\log W(x)$ values, subtract $\log p$ from each array entry corresponding to $x \equiv x_0 \bmod p$. After sieving with all p in the factor base, array entries which are below some threshold are called *candidate relations*, and are checked for smoothness by actually factorising them.

In the number field sieve, the (a, b) pairs for which $F(a, b)$ contains an appearance of the factor base element (p, r) are regularly spaced mod p. Sieving is therefore available as a means of relation collection. As we saw in the previous subsection, if $\mathfrak{p} \leftrightarrow (p, r)$ for some $r \in \mathcal{R}'(p)$ is an ideal in the factor base, then for coprime (a, b) we have

$$p|F(a, b) \;\Leftrightarrow\; a - br \equiv 0 \bmod p.$$

This gives rise to an obvious method of sieving. Start with an array of values $\log F(a, b)$. Fix b and find the first $a = a_0$ (in the relevant range) for which $a_0 \equiv br \bmod p$. Then subtract $\log p$ from the array entries corresponding to a_0 and to the remaining $a \equiv a_0 \bmod p$. Then increment b. This is called *classical sieving* and was the first method suggested for the number field sieve ([14], [63] and [48]).

John Pollard then suggested the improvement which he called *lattice sieving* [64]. An extension was implemented in [35]. Lattice sieving is substantially faster than classical sieving, and has all but replaced it in practice.

The idea of lattice sieving is as follows. Fix a set Q of primes q for which F has at least one root modulo each q. Each $q \in Q$ is called a *special* q. Sieving occurs *only* over those (a, b) for which it is known that $q|F(a, b)$ for some $q \in Q$. If q is not too small, then knowing that $q|F(a, b)$ renders it more likely that $F(a, b)$ is smooth. Clearly, smooth values of $F(a, b)$ without a divisor in Q will be missed, but Q is chosen to ensure that the cost in missed relations is much less than the gain in efficiency.

We receive a gain in efficiency because it is quick to generate, given a factor base element (q, s) with $q \in Q$, the (a, b) pairs for which $q|F(a, b)$ and $a/b \equiv s \bmod p$. Such pairs form a lattice $L_{q,s}$ in the (a, b) plane, so can be generated quickly using a reduced lattice basis.

Within $L_{q,s}$ we continue sieving with each prime $p < q$ which occurs in the factor base. Sieving with p occurs in one of two ways, *by rows* or *by vectors*. Denote by (c, e)

the coordinate system in $L_{q,s}$ with respect to its reduced basis. Sieving by rows fixes e and, analogously to classical sieving, sieves the factor base elements (p, r) with $p < q$ along e. Sieving by vectors regards pairs in the (c, e) plane corresponding to (a, b) pairs for which $p|F(a, b)$ with $a/b \equiv r \bmod p$, as a sub-lattice (abbreviated to $L_{q,p}$) of L_q. A basis for $L_{q,p}$ is not always well-defined. If not, then this (p, r) is sieved by rows. If so, then $L_{q,p}$ is generated from a reduced basis. More complete details of these processes are to be found in [35].

Line sieving (see [33] for details) is similar to lattice sieving by rows. It corresponds to lattice sieving with fixed b. That is, for each special q, fix b then perform lattice sieving on all (a, b) for which $q|F(a, b)$, then increment b. Incrementing b is a comparatively expensive operation. Typically, polynomials generated for use with the line siever are re-written so that b is not changed often. Indeed, often b is fixed at $b = 1$. This is made more efficient by the use of "skewed" bases for the relevant sub-lattices in the (a, b) plane.

For the most part in this thesis we use the number field sieve implementation described in [33]. In Chapter 6 we refer to the siever in this implementation as *the CWI siever*. We also report in Chapter 6 on some sieving performed using the lattice siever of [35], adjusted to use the "skewed" basis representations mentioned above. We refer in Chapter 6 to this adjusted lattice siever as *the AKL siever*.

Remark 2.1.9 To this point we have spoken only of algorithmic, or "software" considerations in sieving. Adi Shamir recently proposed a hardware variation [70]. He proposes an opto-electrical device specifically designed to perform sieving operations. In [70] the device is presented only for MPQS sieving, but adjustment for the number field sieve should not be difficult. The device, called TWINKLE, is only in the conceptual stages at present. If built, it would be a cheap platform on which to perform sieving up to 500–1000 times faster than conventional workstations today.

Notice that improvements like TWINKLE to sieving procedures are still subject to new advances in polynomial selection. Polynomials which have good yield do so independently of the sieving device used. The sieving device determines the amount of wall-clock time required to *detect* that yield.

Large Prime Variations

This time can be reduced considerably by accepting F-values which are almost smooth. Such a variation to sieving is known as a large prime variation.

Large prime variations to MPQS are well known, see for example [49] and [7]. We have two smoothness bounds, B_1 and B_2 with $B_1 < B_2$. During sieving, polynomial values are accepted if they contain at most two so called *large primes* between B_1 and B_2, but are otherwise B_1-smooth. Relations containing no large primes are called *full* *relations*. Relations containing exactly one or two large primes are called *1LP-* and *2LP-relations* respectively. 1LP- and 2LP-relations are referred to collectively as *large prime relations*.

In the same way that the primes at most B_1 need to be "paired-up" to form squares, so do the large primes. In MPQS with two large primes (P^2-MPQS) this is usually thought of as a graph theory problem. Let $G = (V, E)$ be the graph whose vertices are the large primes appearing in the relations and the notional vertex 1, and for which $\{P_1, P_2\} \in E$ exactly when the large primes P_1 and P_2 appear in the same large prime relation. A 1LP-relation is represented by the edge $\{P_1, 1\}$. Finding a fundamental cycle in G corresponds to finding a subset of the large prime relations in which every P_i occurring across the subset does so exactly twice. Hence, the large primes are "squared up" by finding fundamental cycles in G.

In MPQS with only one large prime (P-MPQS), the number of cycles obtained (cycles in P-MPQS are better thought of as "matches") as a function of the number of large prime relations gathered has been analysed ([49], [7], [55]). For P^2-MPQS, this analysis is an open problem. One possibility for insight is the theory of random graphs (for example [60]). Most random graph models assume that the probability of a given edge occuring is uniform across E, however see [42] for a treatment with non-uniform edge probabilities.

Large prime variations to the number field sieve are complicated by the presence of two polynomials, each with their own factor base. With smoothness bounds B_1 and B_2, we consider an (i, j)LP-relation to be one for which F_1 is iLP-smooth and F_2 is jLP-smooth. We refer to such a variation with $i \leq I$ and $j \leq J$ as the number field sieve with $(I + J)$ large primes. In [30] an implementation is presented with $I = J = 2$. In subsequent chapters we deal mainly with each polynomial individually, so we refer just to a particular value being iLP-smooth.

The problem of "squaring up" the large primes is complicated significantly in the number field sieve by the presence of two factor bases. Thinking about the problem in terms of cycle finding in graphs is less instructive than with MPQS. Nevertheless we still refer to a set of large prime relations in which every large prime occurring *in each factor base* does so exactly twice in that factor base, as a cycle.

In [30] the number of cycles obtained as a function of the number of relations gathered is considered. The authors observe smooth growth in the number of cycles initially, followed by a sudden, almost vertical increase. This phenomena has become known as *cycle explosion*. Cycle explosion has been observed in other large factorisations (for example, [23]). Indeed, practitioners now expect cycle explosion for large factorisations, and are able to tell that it is imminent (we do not elaborate on this). However, cycle explosion has not been analysed. Connections to the behaviour of random graphs have again been noted, but the existence of two factor bases complicates matters even more than the unresolved P^2-MPQS case. It may be useful to think of the number field sieve case in terms of a hypergraph whose edges span the distinct factor bases. If so, then it is worth noting that [41] extends Kovalenko's result [42] on non-uniform edge probabilities, to hypergraphs.

Finally we note that in the RSA-140 and RSA-155 factorisations discussed in Chapter 6, the CWI siever has $I = 3$ for the non-linear polynomial F_1, and $J = 2$ for the linear F_2. The AKL siever has $I = J = 2$, as in [30] and [23].

2.1.4 The Matrix Step

Before discussing the matrix step itself, we mention an important pre-computation called *filtering*. Filtering is a process which reduces the amount of data entering the matrix, by removing less useful relations. For example, a relation involving an ideal that does not occur in any other relations, is useless. We do not elaborate on filtering here, see [33]. We do however, emphasize that (especially given the following remarks on troublesome matrix sizes) good filtering strategies are becoming increasingly important. For a description of the filtering strategy applied during the RSA-140 factorisation, see [16] and [15]. For earlier considerations similar to filtering (aimed at reducing the number of relations per cycle amongst the large prime relations) see [26].

We now consider the matrix reduction itself. We are required to reduce a large sparse matrix over $\mathbb{Z}_2$. Peter Montgomery's implementation of the Blocked Lanczos Algorithm over $\mathbb{Z}_2$ described in [33] and [52] addresses the problem. For earlier considerations see [44]. We do not give details of these algorithms.

In practice however, the matrix reduction is becoming a bottleneck. In its present form, the Lanczos implementation runs only on a single machine. The matrices requiring reduction are very large. The problem is exacerbated using the number field sieve, as opposed to MPQS, because the number field sieve requires a factor base for *each* polynomial. In fact, the matrix reduction becomes a major practical issue for further record factorisations. Our improvements to polynomial selection however, have an impact on this problem, as do good filtering strategies.

In the next section we meet the following heuristic guide to the expected size of the matrix for factoring some N_2. If $S(N_2, N_1)$ is the ratio of sieving effort required to factorise N_2 compared to that of N_1, and $M(N_2, N_1)$ is the similar ratio for the relative matrix size, then

$$M(N_2, N_1) \approx \sqrt{S(N_2, N_1)}. \tag{2.8}$$

Extrapolation using the asymptotic run-time estimate $L_N[1/3, (64/9)^{1/3}]$ for sieving gives

$$S(\text{RSA-140}, \text{RSA-130}) \approx 4,$$

so we expect

$$M(\text{RSA-140}, \text{RSA-130}) \approx 2.$$

The RSA-130 matrix had approximately 3.5 million rows and columns. A matrix of the expected size 7 million would be troublesome. However, the RSA-140 matrix had only 4.7 million rows and columns, and $4.7/3.5 \approx 1.3$.

How does this discrepancy between expected and actual size come about? Because of our improvements to polynomial selection used for the RSA-140 factorisation, we found that

$$S(\text{RSA-140}, \text{RSA-130}) \approx 2.$$

From this, we expect

$$M(\text{RSA-140}, \text{RSA-130}) \approx \sqrt{2},$$

which is close to the value obtained. So, better polynomial selection not only decreases sieving time, but helps restrict the size of the matrix.

Note however, that it is only possible to exploit the improved yield to restrict the matrix size if good filtering strategies are in place. For us this is certainly the case. In fact, sieving now continues longer than is strictly necessary, so that there is a bigger pool of relations. From this pool it is hoped that a combination of relations can be chosen by filtering to lead to a smaller (or less dense) matrix.

Further extrapolation using the L-function gives

$$S(\text{RSA-155}, \text{RSA-140}) \approx 7.0.$$

This gives

$$M(\text{RSA-155}, \text{RSA-140}) \approx 2.6,$$

and a matrix for RSA-155 well in excess of 10 million rows and columns. This would be problematic. However, according to the analysis in Chapter 6, we made better use of our new polynomial selection methods for RSA-155 than for RSA-140, and so obtained a better polynomial (relatively speaking) for RSA-155. We can expect this to affect both the sieving effort and the size of the matrix favourably.

Further possibilities regarding the matrix step are discussed in Chapter 7.

2.1.5 The Square Root Algorithm

The square root stage of the number field sieve was initially expected to be a technical difficulty for large N. However, thanks again to Peter Montgomery, the problem is essentially solved.

An early method for extracting the relevant square root is outlined in [14]. The method relies on Hensel lifting. However the integers involved in the last few liftings are very large, so that even with fast multiplication techniques, the time taken to multiply them is prohibitive.

In [22] the author suggests a method based on the Chinese Remainder Theorem, that avoids these large multiplications. However, that method requires $d = \deg F$ to be odd.

Montgomery's method relies on ideal arithmetic (in fact on arithmetic of fractional ideals). That is, it makes use of the fact that the ideal factorisation of each $a - b\alpha$ is known, to reduce the problem to a manageable size. Details can be found in [51] and [33], with some minor variations in [59].

2.2 Smooth Integers

We now focus on the study of smooth integers, particularly from an analytic perspective. Recall from Chapter 1 that we refer to an integer chosen uniformly at random from those at most r as a *random value* i_r. In Chapters 3–6 we make extensive use and abuse of well-known results concerning the asymptotic probability that i_r is $r^{1/u}$-smooth (for fixed $u \geq 1$ as $r \to \infty$). In Section 2.2.1 we survey the relevant results on this probability.

We then consider in Section 2.2.2 smooth integers in the context of the number field sieve. The focus is again on asymptotic considerations. We give an exposition of the asymptotic complexity analysis of the number field sieve. The aim is to understand the connection between the asymptotic run-time of the algorithm and the polynomial values which are required to be smooth.

2.2.1 Smooth Integers Generally

Let $P_j(n)$ denote the j-th largest factor of n. Also for $x, y \in \mathbb{Z}$ let

$$\psi(r, B) = |\{n \in \mathbb{Z}^+ : n \leq r \text{ and } P_1(n) \leq B\}|.$$

For $u \in \mathbb{R}$ with $u \geq 0$ we define

$$\rho(u) = \lim_{r \to \infty} \frac{\psi(r, r^{1/u})}{r} \text{ for } u > 1 \tag{2.9}$$

and $\rho(u) = 1$ otherwise. This is called the *Dickman function*, since Dickman studied in [27] the limit in (2.9). Think of $\rho(u)$ as the asymptotic probability that a random value i_r has its largest prime factor at most $r^{1/u}$. That is, $\rho(u)$ is the asymptotic probability that the random value i_r is B-smooth with $u = (\log r)/\log B$.

The Dickman function is well studied, see [58] for a survey of the results. For our purposes it suffices to note the following. It is well known that the Dickman function satisfies

$$\psi(r, r^{1/u}) = r\rho(u) + \frac{r(1-\gamma)\rho(u-1)}{\log r} + O\left(\frac{r}{\log^2 r}\right), \tag{2.10}$$

where γ is Euler's constant [40]. In our range of interest, the second term in (2.10) contributes to the second significant figure of $\rho(u)$.

We denote by $P(r, B)$ the probability that a random value i_r is B-smooth. Using (2.10) we obtain

$$P(r, B) \approx \rho(u) + (1-\gamma)\frac{\rho(u-1)}{\log r} \tag{2.11}$$

as an approximation to $P(r, B)$ which is adequate for our purposes. We use this approximation throughout Chapters 4–6.

Calculating $\rho(u)$

To do so, we need to be able to calculate $\rho(u)$. The Dickman function satisfies the differential-difference equation

$$u\rho'(u) + \rho(u-1) = 0 \tag{2.12}$$

for $u \geq 1$ [27]. It follows immediately that $\rho(2) = 1 - \log 2$. It also follows that $\rho(u)$ can be computed by numerical integration from

$$\rho(u) = \frac{1}{u}\int_{u-1}^{u} \rho(t)dt,$$

see [77]. This method is also used to compute $\rho(u)$ in [39] and [40].

A more effective method is described in [3]. There it is noted that for integers $l \geq 0$ there exist analytic functions $\rho^{(l)}(u)$ that agree with $\rho(u)$ on the interval $[l-1, l]$. Hence, we may obtain Taylor series expansions on those intervals. Moreover, given the Taylor expansion for $\rho^{(l)}(u)$, the Taylor expansion for $\rho^{(l+1)}(u)$ can be obtained.

So, to calculate $\rho(u)$ on $[l-1, l]$ we use

$$\rho^{(l)}(l-\xi) = \sum_{i=0}^{\infty} c_i^{(l)}\xi^i.$$

For $l = 2$ we have $c_0^{(2)} = 1 - \log 2$ and $c_i^{(2)} = 1/i2^i$ for $i \geq 1$ (see [3]). Otherwise

$$c_i^{(l)} = \sum_{j=0}^{i-1} \frac{c_j^{(l-1)}}{il^{i-j}}$$

for $i > 0$, and

$$c_0^{(l)} = \frac{1}{l-1}\sum_{j=1}^{\infty} \frac{c_j^{(l)}}{j+1}.$$

In [3] it is noted that calculating coefficients $c_j^{(l)}$ for $j = 1, \ldots 55$ is sufficient to calculate $\rho^{(l)}(u)$ to a relative error of about 10^{-17}. Throughout this thesis, when $\rho(u)$ is computed we implicitly use the method of [3].

Remark 2.2.1 New results of Bernstein suggest that computing tight upper and lower bounds on $\psi(r, B)$ may become a viable alternative to computing $\rho(u)$ [4]. Since full details are not yet available, we leave this as a subject of further study.

Generalisations of the Dickman Function

There are several generalisations of the Dickman function. Here we mention some that are useful for analysing the appearance of large prime relations, particularly 1LP- and 2LP-relations. Since we do not make great use of these functions in what follows, we mention them here only briefly.

A thorough analysis of the appearance of relations with up to h large primes requires we know something about the *joint distribution* of the sizes of the $h+1$ largest factors of random values. Let

$$\begin{aligned}\psi(r, B_1, \ldots, B_k) &= \psi_k(r, \mathbf{B}) \\ &= |\{n \in \mathbb{Z}^+ : n \le r \text{ and } P_j(n) \le B_j \text{ for } j = 1, \ldots, k\}|.\end{aligned}$$

Also, let

$$\rho(u_1, \ldots, u_k) = \rho_k(\mathbf{u}) = \lim_{r \to \infty} \frac{\psi_k(r, \mathbf{B})}{r} \tag{2.13}$$

with $u_j = (\log r)/\log B_j$. Vershik investigated in [78] the limit (2.13), and it receives further attention in [10]. Think of $\rho_k(\mathbf{u})$ as the asymptotic joint probability that a random value i_r has its j-th largest prime factor at most r^{1/u_j} for $j = 1, \ldots, k$.

The functions $\rho_2(\mathbf{u})$ and $\rho_3(\mathbf{u})$ are particularly relevant to our 1LP- and 2LP-smoothness considerations. The Taylor series method for computing $\rho(u)$ is extended in [3] to compute $\rho_2(\mathbf{u})$. A further extension is given in [45] to compute $\rho_3(\mathbf{u})$. The details are best left to the references; we make use of these methods only very briefly in Chapter 4.

For the most part, instead of calculating $\rho_2(\mathbf{u})$ and $\rho_3(\mathbf{u})$ we make use of a simplifying assumption. We assume that the appearance of $P_j(n)$ is independent of $P_i(n)$ for $i = 1, \ldots, j-1$. Clearly this is not true, but it does suffice for our purposes.

2.2.2 Smooth Integers and the Number Field Sieve

Recall from Chapter 1 that we define

$$L_x[v, w] = \exp\left[\,(w + o(1))\left(\log x)^v (\log\log x)^{1-v}\right)\,\right].$$

Asymptotically the number field sieve is exciting because it achieves $v = 1/3$ whereas all previous algorithms achieve at best $v = 1/2$. As background to the polynomial selection problem, we should understand how the number field sieve achieves $v = 1/3$ in its run-time estimate. We are concerned only with the time taken by the sieving stage, since this is the rate determining step.

The L-function arises in the analysis of general factorisation algorithms because it is connected to the optimal choice of parameters controlling the appearance of smooth integers. For a fixed smoothness bound, smaller integers are more likely to be smooth than larger integers. Ignoring for the moment the question of how the smooth values are collected, the better general factoring algorithms tend to be those which require smaller integers to be smooth. The following theorem (from [14], see [48] for another version) renders the L-function useful to our analysis by quantifying this relationship.

Theorem 2.2.2 *Let $g(B)$ be a function defined on $B \geq 2$ for which $g(B) \geq 1$ and $g(B) = B^{1+o(1)}$ as $B \to \infty$. Then as $x \to \infty$,*

$$\frac{xg(B)}{\psi(x,B)} \geq L_x[1/2, \sqrt{2}],$$

uniformly for $B \geq 2$. Moreover,

$$\frac{xg(B)}{\psi(x,B)} = L_x[1/2, \sqrt{2}] \tag{2.14}$$

if and only if

$$B = L_x[1/2, 1/\sqrt{2}] \tag{2.15}$$

as $x \to \infty$.

The expression

$$\frac{xg(B)}{\psi(x,B)} \tag{2.16}$$

measures the effort required to find at least B random values i_x which are B-smooth ($g(B)$ is an upper bound on the effort required to test each number for B-smoothness). Theorem 2.2.2 seeks the value of B that minimizes (2.16). The value (2.15) does so, and the minimum value of the required effort is given by (2.14).

Thereom 2.2.2 is also the source of the heuristic guide to relative matrix size given in (2.8). The matrix has approximately B rows and columns. Hence, we can expect a matrix of size approximately (2.15) rows and columns from a sieving effort of approximately (2.14) operations. Observing that

$$L_x[1/2, 1/\sqrt{2}] = \left(L_x[1/2, \sqrt{2}]\right)^{1/2}$$

gives (2.8).

Loosely stated, Theorem 2.2.2 says the following. If $x = x(N)$ is the bound on integers which are required to be smooth by some algorithm A for factoring N, then with an optimal choice of parameters the asymptotic run-time of A is

$$L_x[1/2, \sqrt{2}]. \tag{2.17}$$

The exercise now becomes estimating x as a function of N.

For example, MPQS has $x = O(N^{1/2})$. Substituting this into (2.17) gives

$$\begin{aligned} L_x[1/2, \sqrt{2}] &= \exp\left[\left(\sqrt{2} + o(1)\right) (\log N^{\frac{1}{2}})^{\frac{1}{2}} (\log\log N^{\frac{1}{2}})^{\frac{1}{2}}\right] \\ &= \exp\left[(1 + o(1)) (\log N)^{\frac{1}{2}} (\log(\frac{\log N}{2}))^{\frac{1}{2}})\right] \\ &= \exp\left[(1 + o(1)) (\log N)^{\frac{1}{2}} (\log\log N)^{\frac{1}{2}}\right] \\ &= L_N[1/2, 1], \end{aligned}$$

which is the heuristic asymptotic run-time of MPQS. The bound $x = O(N^{1/2})$ is exponential in $\log N$. By repeating the argument above it is clear that for all such exponential bounds $x = O(N^{1/k})$ with $k > 1$, the run-time is $L_N[1/2, \sqrt{2/k}]$. That is, no exponential bound on x will defeat $v = 1/2$. To do so requires a bound on x which is at worst sub-exponential.

In the number field sieve, a sub-exponential bound on x is achieved. The extent of the sub-exponentiality we again measure using the L-function. The bound on x is

$$x = L_N[2/3, (64/3)^{1/3}]. \tag{2.18}$$

It is a simple matter of substitution to check that (2.18) combined with (2.17) gives the stated run-time for the number field sieve of $L_N[1/3, (64/9)^{1/3}]$. It is not such a simple matter to derive (2.18). We outline this now.

Remark 2.2.3 We assume for consistency with [14] that f_1 is chosen to be the *monic* base-m representation of N. That gives a_i and m both $O(N^{1/d})$. In practice we use non-monic f_1 of course, but this does not affect the asymptotic analysis ($d \to d + 1$ as $d \to \infty$). We drop the assumption when deducing practical guidelines for choosing fixed d in Chapter 3.

We have variables N, d and B already defined, and we introduce U to be the maximum value of $|a|$ and b across the sieve region. The idea of the analysis is that we first deduce (using Theorem 2.2.2), for fixed N and d, an asymptotic run-time for optimal choices of U and B. Then we choose a degree which minimizes, as a function of N as $N \to \infty$, this run-time. The choices of U and d fix the size of the integers inspected for smoothness.

In essence, the integers inspected for smoothness are forced to be sub-exponential in $\log N$ by increasing d, very slowly, as a function of N. The compromise for d is between a high rate of change of f at high d, and large coefficients at low d.

The main steps are as follows. The values required to be smooth are bounded by

$$x(N) = F_1(a, b) \cdot F_2(a, b) \leq (d + 1)m^2 U^{d+1} \leq 2dm^2 U^{d+1}. \tag{2.19}$$

Given Remark 2.2.3, (2.19) gives

$$F_1(a,b) \cdot F_2(a,b) \leq 2dN^{2/d}U^{d+1}. \tag{2.20}$$

Assume for the moment that values $F_1(a,b) \cdot F_2(a,b)$ are as likely to be smooth as random integers of the same size. In practice we will rely on this not being true, but the assumption suffices asymptotically. Using this assumption, Theorem 2.2.2, and (2.20), it is shown in [14] that optimal choices of B and U ensure that the asymptotic run-time is

$$\exp\left[(1+o(1))\left(d\log d + \sqrt{(d\log d)^2 + 4\log(N^{1/d})\log\log(N^{1/d})}\right)\right]. \tag{2.21}$$

Now we need to choose d to minimize (2.21). As is pointed out in [14], the minimum value of (2.21) must occur when

$$(d\log d)^2 \;=\; O\left(\log(N^{1/d})\log\log(N^{1/d})\right). \tag{2.22}$$

Intuitively, it is clear that d ought to have the form

$$O\left((\log N)^j(\log\log N)^k\right) \tag{2.23}$$

since the appearance of d on the left hand side of (2.22) must match the form of the right hand side. Assuming d is of that form and ignoring for the moment the implicit constants, gives

$$\begin{aligned} (d\log d)^2 &\approx (\log N)^{2j}(\log\log N)^{2(k+1)}, \text{ and} \\ \log(N^{1/d})\log\log(N^{1/d}) &\approx (\log N)^{1-j}(\log\log N)^{1-k}, \end{aligned}$$

from which we obtain $j = 1/3$ and $k = -1/3$. Substituting these values yields the constant implicit in (2.23).

The result is that the optimal value of d as $N \to \infty$ is

$$d = \left(3^{1/3} + o(1)\right)\left(\frac{\log N}{\log\log N}\right)^{1/3}. \tag{2.24}$$

At fixed N in our range of interest, some information is lost in the approximations and assumptions leading to (2.24). For the asymptotic result however, (2.24) is useful; substituting the optimal value of d back into (2.21) gives the $L_N[1/3,(64/9)^{1/3}]$ estimate. Substituting the optimal values of U and B into (2.20) leads, after some manipulation, to (2.18).

Hence, in essence, the number field sieve defeats other algorithms asymptotically because the size of the values required to be smooth is sub-exponential in $\log N$. This is guaranteed asymptotically by controlling, through d, the size of the relevant polynomial values, thereby encouraging the polynomials to produce more smooth values.

Leveraging this advantage in practice requires using polynomials which do indeed output many smooth values. Asymptotically the advantage comes from increasing d as $N \to \infty$. Differences in yield between polynomials of fixed degree do not affect the asymptotics. Indeed, not even the difference between monic and non-monic F_1 affects the asymptotics. We should not ignore what is revealed in the asymptotics, neither will we ignore what is hidden.

2.3 Polynomial Selection

From Sections 2.1 and 2.2 we understand that (and why) smooth polynomial values are crucial to the performance, in practice and asymptotically, of the number field sieve. So we arrive at the polynomial selection problem. That is, given N, how do we find polynomial pairs for N which produce many smooth values ?

We distinguish two aspects of this problem; generating candidate pairs at all, and generating good candidate pairs. We saw in Section 2.1 that to ensure the squares obtained from smooth values of F_1 and F_2 are congruent mod N, f_1 and f_2 must have a common root mod N. This requirement makes the first aspect, generating candidate pairs at all, non-trivial. Our focus in this thesis is on the second aspect, generating good candidate pairs. We will assume procedures for generating pairs satisfying the relevant requirements. The main purpose of this section is to describe such procedures, that is, procedures addressing the first aspect.

There is an obvious method for generating f_1, f_2 with a common root mod N. It is called the *base-m method*, and was suggested for use in the number field sieve in [14]. We saw this method briefly in Section 1.3.3. With $d = \deg F_1$ fixed in advance and $m = O(N^{1/(d+1)})$, the coefficients of (non-monic) f_1 are taken from the base-m expansion of N and are therefore expected to satisfy $a_i = O(N^{1/(d+1)})$. Then f_2 is the linear polynomial $x - m$. The base-m method is restrictive, in that m *must* be small to keep f_2 and the coefficients of f_1 small. In general, there may exist many more pairs f_1, f_2 with an *arbitrarily large* common root m mod N. Since both such polynomials are likely to be non-linear, we refer to methods giving these polynomials as *non-linear methods*.

In Section 2.3.1 we describe what little known is about non-linear selection methods. It transpires that the base-m method is still the best known method for large N. We give more background on the base-m method in Section 2.3.2.

2.3.1 Non-linear Methods

The first step in non-linear selection methods is an algorithm due to Peter Montgomery, reported in [33]. It finds pairs of quadratic polynomials with a common root m mod N, each of whose coefficients are $O(N^{1/4})$. Analysis of the algorithm reveals that this

$O(N^{1/4})$ is $O(N^{1/2d})$ at $d = 2$. We call this method *Montgomery's Two Quadratics method.*

Let $(d_1, d_2) = (\deg f_1, \deg f_2)$, and $d_T = d_1 + d_2$. We refer to (d_1, d_2) as the *degree pair* for f_1, f_2. Montgomery's Two Quadratics method gives the degree pair $(2, 2)$. Since $d_T = 4$, the comparable base-m pair is $(3, 1)$. For the integers we consider in Chapter 6, $(5, 1)$ is the appropriate base-m degree pair. Whilst we might expect two quadratic polynomials to be competitive with cubic base-m pairs, we cannot expect a pair of quadratic polynomials to be competitive beyond, say, integers of length 110–120 digits.

There are however, prospects of extending Montgomery's Two Quadratics method to higher degrees. In particular, we seek two polynomials each of degree d and each of whose coefficients are $O(N^{1/2d})$. We are most interested in the degree pair $(3, 3)$, since this may be competitive with $(5, 1)$ base-m polynomial pairs. Below we describe Montgomery's Two Quadratics method, and possibilities for extension.

Montgomery's Two Quadratics Method

The description given here is essentially reproduced from [33] with some details omitted. Suppose we have two quadratic polynomials $f_1(x) = a_2x^2 + a_1x + a_0$ and $f_2(x) = b_2x^2 + b_1x + b_0$ in $\mathbb{Z}[X]$. Let

$$\mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}.$$

The key observation is this: f_1 and f_2 have a common root m modulo N if and only if $\mathbf{a}$ and $\mathbf{b}$ are orthogonal (over $\mathbb{Z}_N$ with respect to the standard inner product) to the vector

$$\mathbf{c} = \begin{bmatrix} 1 \\ m \\ m^2 \end{bmatrix}.$$

The elements of $\mathbf{c}$ form a geometric progression over $\mathbb{Z}_N$ with ratio m. The space orthogonal to $\mathbf{a}$ and $\mathbf{b}$ has rank 1 (see [33]), therefore any $\mathbf{c}$ in that space whose elements are in the same progression will suffice to generate the space.

So, Montgomery begins with such a vector $\mathbf{c}$, and then constructs a basis for the space orthogonal to $\mathbf{c}$. Indeed, if p is a prime such that $p < \sqrt{N}$, the Legendre symbol $(N/p) = 1$, and c_1 a square root $\bmod\, p$ of N with $|c_1 - N^{1/2}| \leq p/2$, then

$$\mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} p \\ c_1 \\ (c_1^2 - N)/p \end{bmatrix}$$

is a suitable $\mathbf{c}$ with $c_i = O(N^{1/2})$. The ratio of the elements of $\mathbf{c}$ over $\mathbb{Z}_N$ is $m = c_1p^{-1} \bmod N$. The multiplication by $p^{-1} \bmod N$ is what causes m to take arbitrarily large values $\bmod\, N$.

The following vectors $\mathbf{a}'$ and $\mathbf{b}'$ are both orthogonal over $\mathbb{Z}_N$ to $\mathbf{c}$, and in fact span the sub-lattice of $\mathbb{Z}^3$ orthogonal to $\mathbf{c}$. We have

$$\mathbf{a}' = \begin{bmatrix} c_1 \\ -p \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{b}' = \begin{bmatrix} (c_1(c_2 s \bmod p) - c_2)/p \\ -(c_2 s \bmod p) \\ 1 \end{bmatrix}.$$

By reducing the basis $\{\mathbf{a}', \mathbf{b}'\}$ a basis $\{\mathbf{a}, \mathbf{b}\}$ can be found for which

$$\|\mathbf{a}\| \cdot \|\mathbf{b}\| = O(\|\mathbf{c}\|) = O(N^{1/2}).$$

In practice both $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$ are $O(N^{1/4})$. Each p gives a distinct pair of polynomials, so what remains is to search amongst many pairs of polynomials, for the best ones.

Remark 2.3.1 The number field sieve can be generalised to use k polynomials all with a common root m mod N. An early but highly theoretical suggestion along these lines is contained in [20]. A more practical version is suggested in [32]. Crucial to the success of this suggestion is an adequate means of polynomial selection. The implementation described in [32] uses Montgomery's Two Quadratics method with small coprime linear combinations of the polynomials found. We refer to this scheme again briefly in Chapters 4 and 7.

Extensions to Higher Degree

The fact which endears quadratic polynomials to Montgomery's construction is that the space orthogonal to $\mathbf{a}$ and $\mathbf{b}$ has rank 1. In general, we desire two polynomials of degree d with coefficient vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}^{d+1}$. The space orthogonal to $\mathbf{a}$ and $\mathbf{b}$ has rank $d-1$. So now we need $d-1$ polynomials whose coefficient vectors (of length $d+1$) are mutually orthogonal to the same geometric progression mod N.

Montgomery suggests generating them in the following manner [53]. Suppose we begin with a single vector $\mathbf{c} \in \mathbb{Z}^{2d-1}$ whose coefficients are in geometric progression modulo N. The $d-1$ coefficient vectors are now read off from $\mathbf{c}$; they are precisely the sequences of length $d+1$ consisting of consecutive elements of $\mathbf{c}$. We need to enforce some restriction on $\|\mathbf{c}\|$, to control $\|\mathbf{a}\| \cdot \|\mathbf{b}\|$. We again assume that in practice $\|\mathbf{a}\| \approx \|\mathbf{b}\|$. That being the case we need $c_i = O(N^{1-1/d})$ to ensure that $a_i \approx b_i = O(N^{1/2d})$ if $\mathbf{a}$ and $\mathbf{b}$ are constructed in this way.

Hence, polynomials with (d, d) degree pairs would follow from the construction of small geometric progressions $\mathbf{c} \in \mathbb{Z}^{2d-1}$ mod N (where small means each $c_i = O(N^{1-1/d})$). At $d = 3$, we require geometric progression mod N of length 5 with each $c_i = O(N^{2/3})$. Initial experiments and counting arguments suggested that for large N, such progressions could be difficult to find. Hence we do not pursue this here (see Chapter 7).

Notice that the “small geometric progressions” construction gives $\deg f_1 = \deg f_2$. Other combinations may be preferable, for example with $d_T = 6$ the pair $(4, 2)$ could be useful. Methods to generate such polynomials are not known.

2.3.2 The Base-m Method

For large N therefore, the base-m method is still the method of choice. The base-m method is very simple, here we describe it and some existential arguments in the literature concerning approximately optimal choices of such polynomials.

Let the coefficients of the base-m representation of N be $a_i^{(m)}$. That is,

$$N = \sum_{i=0}^{d} a_i^{(m)} m^i.$$

with $0 \leq a_i^{(m)} < m$. More generally, the base-m representation of kN for some small $k \in \mathbb{Z}$ can be taken. For ease of expositition we assume that just the representation of N is taken.

It is not necessary that the polynomial $F(x, 1) = f(x)$ be the true base-m expansion of N, simply that

$$f(m) \equiv 0 \bmod N. \tag{2.25}$$

Any alteration can be made to the coefficients of f provided the property (2.25) is preserved. An alteration which leaves f with smaller coefficients is, at least heuristically, useful. In particular, if $a_i > \lfloor m/2 \rfloor$ then making the replacements

$$\begin{aligned} a_i &\mapsto a_i - m \quad \text{and} \\ a_{i+1} &\mapsto a_{i+1} + 1 \end{aligned} \tag{2.26}$$

leaves the representation with smaller a_i, whilst preserving (2.25). We assume below that f has been reduced in this way, working from $i = 0, \ldots, d$ through the coefficients. Note that some authors perform a LLL reduction [46] on the lattice created by transformations of this type in the hope of finding slightly smaller coefficients (see [80] and [85]). We omit this computation and proceed simply with the adjustment at (2.26). More details of our procedures emerge in Chapter 5.

A slight variation on the base-m method is suggested in [14]. The suggestion uses the full “homogeneity” of F_1 and F_2. That is, instead of fixing m as a root mod N of $f_1(x)$, fix (m_1, m_2) as a root mod N of $F_1(x, y)$. Then $F_2(x, y)$ is given by $F_2(x, y) = m_2 x - m_1 y$. The advantage in doing so is that some of the size borne by a single value m can be shared across m_1 and m_2, and so across the coefficients of F_2. Essentially, this is the analogue for F_2 of using non-monic F_1.

Remark 2.3.2 Unfortunately, choosing good base-m polynomials is already a difficult problem, so choosing good base-(m_1, m_2) polynomials has received little attention in the literature. It is noted in [5] however that this method could be re-considered should improvements be made to the choice of base-m polynomials that make it worthwhile (see Chapter 7).

For the moment then, we are stuck with (possibly modified) base-m polynomials. How good can we expect these polynomials to be? There is some discussion on this question in [14] and [5]. Next we summarise these discussions, with some adjustments.

Suppose $\max |a_i| \leq A$ and $m \leq M$. We have two requirements of F_1 and F_2; that F_1 and F_2 have a common root mod N and that this should hold *for all* integers $1, \ldots, N$. (The argument that follows can be adjusted for the case that the common root is required only *for some* positive $n \leq N$, and this corresponds to the special number field sieve). The common root requirement means that, in particular, $f_1(m) \geq N$, which gives

$$O(AM^d) \geq N. \tag{2.27}$$

The requirement that this hold for all integers $1, \ldots, N$ means that the number of integers representable by the possible f_2 at possible m must exceed N. That is,

$$O(A^{d+1}M) \geq N. \tag{2.28}$$

Let $A = N^{\mu}$ and $M = N^{\nu}$. In [14] and [5] the argument is that for fixed N and d, some guidance can be obtained on optimal values for μ and ν by requiring (2.27) and (2.28) to be equalities. Doing so, and solving for μ and ν, gives

$$\mu = \frac{d-1}{d^2+d-1} \quad \text{and} \quad \nu = \frac{d}{d^2+d-1} \ .$$

Notice that these values differ from those in [14], because there the authors consider the prospects for base-(m_1, m_2) polynomials, whereas we do not yet permit non-monic f_2.

For example, with $N =$ RSA-140, we can hope to obtain at best $A \leq 10^{19.3}$ if $M \leq 10^{24.1}$. That is, coefficients approximately five digits smaller than m. In effect, our methods achieve this. As we see in Chapter 6, using root properties we can effectively shave up to three digits from the coefficients of F_1. Simultaneously, we save approximately up to two digits by having regard to the size of the values taken by F_1. Now we investigate how this comes about.

Chapter 3

Properties which Influence Yield

In this chapter and the next we study polynomial yield. This chapter establishes a framework by parameterising the effects of the relevant properties. The next chapter uses this framework to investigate the influence of the properties more thoroughly.

As noted in Chapter 1, there are two factors which influence the yield of a given number field sieve polynomial F_1. We call the factors size and root properties. By *size* we refer to the magnitude of the values taken by F_1. By *root properties* we refer to the distribution of the roots of F_1 modulo small p^k, for p prime and $k \geq 1$. We are interested in the effect of root properties on the likelihood of F_1 values being smooth. In short, if F_1 has many roots modulo small p^k, values taken by F_1 "behave" as if they are smaller than they actually are. That is, on average, the likelihood of F_1 values being smooth is increased.

It has always been well understood that size affects the yield of F_1. The influence of root properties however, has not previously been either well understood or adequately exploited. Hence in this chapter we focus more on root properties than size.

In Section 3.1 however we do consider briefly an issue regarding size that is peculiar to the number field sieve. In particular, we discuss the choice of polynomial degree d for given N.

In Section 3.2 we lay the foundations for quantifying the effect of root properties. Recall from Section 1.4.1 that we use the term *random value* i_r to refer to an integer chosen uniformly at random from $\{i \in \mathbb{Z} : 1 \leq i \leq r\}$. Here, the effect of root properties is quantified by comparing polynomial values v to random values i_r with $r = v$. We give a heuristic estimate of the expected contribution of each prime p to each value. That is, we estimate the average exponent of p appearing in a sample of factorisations. The expected contributions of each p are different for F-values compared to random values. This gives a means of assessing the "behaviour" of the typical F-value compared to a random value of the same size. From this, we quantify the effect of root properties.

This is an adaptation of an approach used in the analysis of the continued fraction method and of MPQS, which we discuss briefly in Section 3.2. We then calculate contributions of p in some relevant cases, check empirically the validity of the estimations, and so deduce our parameter $\alpha(F)$ which is used to quantify the effect of root properties. Finally, we consider root properties with respect to the degree of F. In the quadratic case, we demonstrate the significance of attention to root properties. We also examine the average root structure for polynomials of higher degrees.

Section 3.3 contains a summary of this chapter.

3.1 Size

The manner in which size influences yield is clear. We saw in Section 2.2.1 that the smoothness probability of random values i_r, as a function of r, is well understood. Hence, given N and d, the exercise in choosing (F_1, F_2) with good size is clear: the size of the values over which sieving is to occur should be kept small.

However, d of course is not fixed initially. We saw in Section 2.2.1 that the key to the asymptotic performance of the number field sieve is that the degree of F_1 is optimised to minimize the run-time of sieving. These however, are asymptotic considerations. Here we consider in more detail, what is the best choice of d for N in the current range of interest and for the polynomials we currently use.

First we recall some details from Section 2.2.2. Assume that we are working with base-m polynomials. So $F_1(x, y)$ is the non-linear polynomial and $F_2(x, y)$ the linear polynomial. If U is an upper bound for the values $|a|$ and b defining the sieve region, then

$$F_1(a, b) \cdot F_2(a, b) \leq 2dm^2U^{d+1} \tag{3.1}$$

is an upper bound on the values inspected for smoothness during sieving. Assuming that F_1 is monic, and therefore that $m \approx N^{1/d}$, this gives

$$F_1(a, b) \cdot F_2(a, b) \leq 2dN^{2/d}U^{d+1}. \tag{3.2}$$

Using (3.2), it is shown in [14] that optimal choices of U and B ensure the run-time of the number field sieve, with d and N fixed does not exceed

$$\exp\left((1 + o(1))\left(d \log d + \sqrt{(d \log d)^2 + 4 \log(N^{1/d}) \log\log(N^{1/d})}\right)\right). \tag{3.3}$$

Choosing d to minimize (3.3) gives the optimal choice of d *asymptotically* as

$$d = \left(3^{1/3} + o(1)\right)\left(\frac{\log N}{\log\log N}\right)^{\frac{1}{3}}. \tag{3.4}$$

As $d \to \infty$, which it does very slowly, (3.4) is a useful indication of the appropriate value of d. But at small d, some of the approximations leading from (3.3) to (3.4) may be misleading. Moreover, (3.3) carries the assumption from (3.2) that F_1 is monic. Whilst this makes no difference asymptotically, it may affect the ranges of appropriate

d when d is small. Hence, below we re-write (3.3) for the non-monic case and consider the new expression for small d and for N in in the range of interest.

Using non-monic F_1, the upper bound deduced from (3.1) becomes

$$F_1(a,b) \cdot F_2(a,b) \leq 2dN^{2/(d+1)}U^{d+1}. \tag{3.5}$$

Using (3.5) in place of (3.2) and repeating the argument from [14] which leads to (3.3) gives that the time taken for the number field sieve to factorise N is at worst given by

$$\exp\left((1+o(1))\left(d\log d + \sqrt{(d\log d)^2 + 4\log(N^{1/(d+1)})\log\log(N^{1/(d+1)})}\right)\right), \tag{3.6}$$

using a value for d which minimizes the expression.

That is, to factorise N we should use d which minimizes

$$\mathcal{E}(d,N) = d\log d + \sqrt{(d\log d)^2 + 4\log(N^{1/(d+1)})\log\log(N^{1/(d+1)})}\ .$$

Table 3.1 gives values of $\mathcal{E}(d,N)$ for d and N in the range of interest. The values N_i in the table are the integers 10^{i-1}, that is, integers with i digits. For each N_i the optimal value of d is bolded. For the purposes of illustration, the table begins with integers of length 80 digits. Beware that for integers up to, say, 110 digits long, Montgomery's Two-Quadratics method may be preferable to the base-m method.

Table 3.1 shows that the relevant degrees are $d = 4, 5, 6$. The cut-off between $d = 4$ and $d = 5$ is at approximately 120 digits. The cut-off between $d = 5$ and $d = 6$ is at approximately 220 digits (these figures of course should be used only as a rough guide).

Remark 3.1.1 We have considered only base-m polynomials in this section. That is, we have pairs of polynomials (F_1, F_2) with degree pair $(d, 1)$. It is entirely possible, in fact probably true, that other *combinations* of degree are preferable. For example, the degree combination $(2, 4)$ is likely to be preferable to $(5, 1)$. As noted in Section 2.3.1, since we do not presently know how to generate such polynomial pairs with a common root mod N, we do not consider such possibilities here.

3.2 Root Properties

We turn now to the main concern of this chapter, root properties. In Section 3.2.1 we explain the model we use to quantify root properties, the so-called typical F-value model. In Section 3.2.2 we estimate in general the key quantity in this model. In Section 3.2.3 we use this estimate to construct the parameter $\alpha(F)$ which quantifies the average effect of root properties in F-values. That completes the parameterisation of root properties. In Section 3.2.4 we go on to consider the effect on root properties of varying d, for the relevant cases $d = 4, 5, 6$.

i	$\mathcal{E}(3, N_i)$	$\mathcal{E}(4, N_i)$	$\mathcal{E}(5, N_i)$	$\mathcal{E}(6, N_i)$	$\mathcal{E}(7, N_i)$
80	29.85	**29.08**	29.92	32.06	35.29
90	31.89	**30.83**	31.44	33.36	36.38
100	33.83	**32.51**	32.90	34.61	37.44
110	35.69	**34.13**	34.30	35.83	38.48
120	37.49	35.68	**35.67**	37.01	39.50
130	39.21	37.19	**36.99**	38.17	40.50
140	40.89	38.65	**38.27**	39.29	41.48
150	42.52	40.06	**39.51**	40.39	42.44
160	44.09	41.44	**40.73**	41.47	43.38
170	45.63	42.78	**41.92**	42.52	44.30
180	47.13	44.09	**43.08**	43.54	45.21
190	48.59	45.37	**44.21**	44.55	46.10
200	50.02	46.62	**45.32**	45.54	46.98
210	51.42	47.84	**46.41**	46.51	47.85
220	52.79	49.04	**47.47**	**47.47**	48.70
230	54.14	50.22	48.52	**48.40**	49.53
240	55.46	51.38	49.55	**49.33**	50.36
260	58.03	53.63	51.56	**51.13**	51.98
280	60.51	55.81	53.50	**52.88**	53.55
300	62.92	57.92	55.39	**54.58**	55.08

Table 3.1: $\mathcal{E}(d, N)$ at relevant d and N

3.2.1 The Typical F-value

Ideas similar to the "typical F-value" analysis presented here for the number field sieve, have previously been introduced for analysis of MPQS [6] and the continued fractions method [39]. The Knuth-Schroeppel analysis of [39] examines the use of small multipliers k in the continued fractions method, Boender's analysis in [6] extends this in the context of small multipliers for MPQS.

The situation regarding small multipliers is similar for the continued fractions method to that for MPQS, so we elaborate only on MPQS. In MPQS, kN for some small k may be a quadratic residue for more small p than N is. The benefit is an increased likelihood that values of the relevant quadratic polynomial will be smooth, the cost is that now the integer to be factorised is larger. Hence, analysis is required to choose k so that the benefit exceeds the cost, hopefully optimally.

The idea which we distil from [39] and [6] is that it is useful to examine the quantity which we refer to as $\mathrm{cont}_p(v)$.

Definition 3.2.1 *Denote by* $\mathrm{ord}_p v$ *the exponent of the largest power of* p *dividing* v.

Then $\mathrm{cont}_p(v)$ *is the expected value of* $\mathrm{ord}_p v$ *as* v *ranges across some sample* $\mathcal{S}$*.*

So, on average, $v \in S$ looks (across the primes at most B) like

$$\log v = \sum_{p \le B} \mathrm{cont}_p(v) \cdot \log p \ . \tag{3.7}$$

In the special case where S is a set of F-values v, we denote $\mathrm{cont}_p(v)$ by $\mathrm{cont}_p(F)$ and refer to the exponential of the value in (3.7) as the *typical F-value.*

In [39] $\mathrm{cont}_p(kN)$ is called $f(p, kN)$. In [6] a comparison is made between $\mathrm{cont}_p(v)$ for values v of quadratic MPQS polynomials and for random values $v = i_r$ (although in [6] the terminology is different). Here we make a similar comparison; we compare the typical F-value, for number field sieve polynomials, to the typical random value.

Notice that $\mathrm{cont}_p(v)$ for $v \in S$ is easy to check empirically. For sufficiently large S we expect

$$\mathrm{cont}_p(v) \approx \frac{\sum_{v \in S} \mathrm{ord}_p v}{|S|} \ . \tag{3.8}$$

In particular, for F-values, $\mathrm{cont}_p(F)$ can be determined by factorising a small, but not too small, set of F-values in the appropriate range. For most p however, we can do better by giving a heuristic explicit form for $\mathrm{cont}_p(F)$. The primes p for which we can do this are precisely those for which we can assume that the full contribution of p in a given F-value is associated with a single contribution from a single root mod p of F. That is, the primes p which are unramified. Ramified primes must divide Δ, the discriminant of f (see for example [18]). As a coarse filter on ramified primes, we refer to p for which $p|\Delta$ as *poorly-behaved primes*, otherwise p is *well-behaved.*

In the following subsection we give heuristic estimates of cont_p in the relevant cases, for well-behaved primes. Contributions of poorly-behaved primes p could be obtained by computing the ideal decomposition of $\langle p \rangle$([18] Section 6.2). However, we find in practice it is simpler to compute these contributions directly from a sample of factorisations, as in (3.8).

3.2.2 Estimating $\mathrm{cont}_p(v)$

It is useful to distinguish three cases; the random value i_r, polynomial values of the form $F(x, 1) = f(x) = v$ and polynomial values of the more general form $F(x, y) = v$. We develop estimates of $\mathrm{cont}_p(v)$ in these cases from ideas due to Peter Montgomery.

Consider a random value i_r. It is possible that powers p^k for integer $k > 1$ also divide i_r, so we expect p to appear as

$$p^{\left(\frac{1}{p} + \frac{1}{p^2} + \dots\right)} = p^{\frac{1}{p-1}}$$

in i_r. Hence, we take the average contribution of p to i_r to be

$$\mathrm{cont}_p(i_r) = \frac{1}{p-1} \ . \tag{3.9}$$

Even though (3.9) is merely a heuristic estimate it works well in practice. Table 3.2 shows estimated and actual contributions of $p < 50$ in a total of 10^5 integers chosen uniformly at random in the interval $[10^{20}, 10^{21}]$.

p	Actual	Estimate
2	100213	100000
3	50280	50000
5	25062	25000
7	16808	16667
11	10118	10000
13	8196	8333
17	6202	6250
19	5529	5556
23	4590	4545
29	3629	3571
31	3333	3333
37	2786	2778
41	2446	2500
43	2401	2381
47	2263	2174

Table 3.2: Actual and expected contributions of p in i_r for $p < 50$

Consider polynomial values of the form $f(x) = F(x, 1) = v$. These values are of use both as an easier version of the more general case, and are of interest in their own right when examining line sieving. Now, since each root mod p corresponds to a unique root mod higher powers of p by Hensel lifting, the full contribution from each root is $1/(p-1)$. Think of each root mod p as a distinct opportunity for an f-value to be divisible by p. If there are q_p distinct roots of f mod p then we take the full contribution of p to the typical f-value to be

$$\text{cont}_p(f) = \frac{q_p}{p-1} \; . \tag{3.10}$$

Computational evidence for (3.10) being a good estimate appears after we discuss the next case.

Consider now polynomial values of the form $F(x, y)$ for coprime x and y. We no longer have a unique correspondence between roots of $F(x, 1)$ mod p and roots mod p^k for $k > 1$. Moreover, an extra class of roots emerges from the possibility that $p|y$. Indeed, if also $p|a_d$ then $p|F(x, y)$ since F is homogeneous. We call these roots *projective roots.*

Let q_p now be the number of roots mod p of $F(x, y)$. That is, q_p includes the roots x/y of $F(x, 1)$ mod p *and* projective roots. The full contribution of p to the typical value $F(x, y) = v$ with x and y coprime is given by

$$\mathrm{cont}_p(F) = q_p \frac{p}{p^2 - 1} \ . \tag{3.11}$$

To see this, we will count the contribution of p^k for some fixed $k \in \mathbb{Z}^+$, then sum these contributions over k.

Since F is homogeneous, think of the coprime pairs (x, y) as points on the projective line. For the purposes of counting the different combinations of x and y it is useful to consider classes of points as follows.

There are three cases, labelled "s", "0", "∞":

1. **Case 1:** $x/y =$ "s" for some $s \in \mathbb{Z}_{p^k}$ with $s \not\equiv 0 \bmod p$ (that is, neither x nor y are divisible by p).
2. **Case 2:** $x/y =$ "0", that is, $x \equiv 0 \bmod p$.
3. **Case 3:** $x/y =$ "∞", that is, $y \equiv 0 \bmod p$.

Now count the number of classes " " which fall into each case.

1. **Case 1:** There is one class in Case 1 for each $s \in \mathbb{Z}_{p^k}$ not divisible by p, so there are $\varphi(p^k) = p^{k-1}(p-1)$ classes in Case 1.
2. **Case 2:** There is one class in Case 2 for each value $x \in \mathbb{Z}_{p^k}$ divisible by p, so there are p^{k-1} classes in Case 2.
3. **Case 3:** Similarly to 2, there are p^{k-1} classes in Case 3.

So there are a total of $p^{k-1}(p+1)$ classes from Cases 1–3.

Each class has the same number of points (x, y) contributing to it:

1. **Case 1:** For fixed s and given some $x \in \mathbb{Z}_{p^k}$ not divisible by p, y is uniquely determined. So there are $\varphi(p^k)$ pairs contributing to each class (the class is determined by s) in Case 1.
2. **Case 2:** For a fixed value of $x \equiv 0 \bmod p$, y may take any invertible value in $\mathbb{Z}_{p^k}$, so there are $\varphi(p^k)$ pairs contributing to each class (the class is determined by x) in Case 2.
3. **Case 3:** Similarly to 2, there are $\varphi(p^k)$ pairs contributing to each class in Case 3.

Hence, a coprime pair $(x, y) \in \mathbb{Z}_{p^k} \times \mathbb{Z}_{p^k}$ selected uniformly at random will fall into a particular class with probability

$$\frac{1}{p^{k-1}(p+1)}.$$

Precisely q_p of these classes correspond to roots mod p of $F(x, y)$ so the probability that such a pair actually contributes p^k is

$$\frac{q_p}{p^{k-1}(p+1)}.$$

Of course, one p-th of these contributions will be counted again when we count the contribution from p^{k+1}, so the contribution only from p^k is

$$\frac{q_p}{p^{k-1}(p+1)}\left(1-\frac{1}{p}\right).$$

Logarithmically, p^k contributes k appearances of p, so we take the full contribution of p to be given by

$$\begin{aligned}
\mathrm{cont}_p(F) &= \sum_{k=1}^{\infty} \frac{kq_p}{p^{k-1}(p+1)}\left(1-\frac{1}{p}\right) \\
&= \frac{q_p}{(p+1)}\left(1-\frac{1}{p}\right)\sum_{k=1}^{\infty}\frac{k}{p^{k-1}} \\
&= \left(1-\frac{1}{p}\right)\left(1-\frac{1}{p}\right)^{-2}\frac{q_p}{p+1} \\
&= q_p\frac{p}{p^2-1}\ .
\end{aligned}$$

Computational evidence for the estimates (3.10) and (3.11) is given in Table 3.3. This table contains estimated and actual contributions of $p < 100$ for well-behaved primes of a particular polynomial P_{11}. Primes p at which F has no roots are omitted. Polynomial P_{11} is a polynomial considered for the factorisation of RSA-130 (see Section 6.1.1 and Appendix B). We considered 10^4 values of P_{11}. Only $p = 2$ is not well-behaved for P_{11}. We have repeated these counts on many polynomials and these results are typical.

We conclude that estimates (3.9), (3.10) and (3.11) are good estimates of cont_p for well-behaved primes in each case.

3.2.3 Quantifying Root Properties

It is now possible to make a comparison between F-values v and random values i_r with $v = i_r$.

During sieving, notionally the full contribution of each prime $p \leq B$ is removed from each value being sieved. In fact we start with the log of the value and subtract the log of each contribution. So after sieving a random value i_r would appear as

$$\log i_r - \sum_{p\leq B}\frac{\log p}{p-1}\ . \qquad (3.12)$$

p	q_p	$10^4 \cdot$ $\mathrm{cont}_p(F)$	$10^4 \cdot pq_p/$ (p^2-1)	$10^4 \cdot$ $\mathrm{cont}_p(f)$	$10^4 \cdot q_p/$ $(p-1)$
3	1	3882	3750	5002	5000
13	2	1549	1548	1667	1667
19	1	530	528	556	556
23	1	435	436	455	455
29	2	687	690	715	714
37	1	259	270	277	278
47	1	215	213	218	217
53	3	568	566	576	577
61	2	328	328	334	333
73	5	687	685	695	694
79	1	126	127	129	128
89	1	115	112	115	114
97	2	206	206	208	208

Table 3.3: Actual and expected contributions of $p < 100$ for P_{11}

Each polynomial value $F(x,y) = v$ or $f(x) = v$ after sieving appears as

$$\log v - \sum_{p \leq B} \mathrm{cont}_p(v) \cdot \log p \ . \tag{3.13}$$

In each case we call the difference between (3.12) and (3.13) the paramater α, so we have

$$\alpha = \sum_{p \leq B} \left[\frac{1}{p-1} - \mathrm{cont}_p(v) \right] \log p \ .$$

Over the well-behaved primes, (3.10) and (3.11) give

$$\begin{aligned} \alpha(f) &= \sum_{p \leq B} (1 - q_p) \frac{\log p}{p-1}, \quad \text{and} \\ \alpha(F) &= \sum_{p \leq B} \left(1 - q_p \frac{p}{p+1} \right) \frac{\log p}{p-1}. \end{aligned}$$

Hence, for example in the latter case we have

$$\log F(x,y) = \log i_r + \alpha(F)$$

whereby we consider F-values to behave like random integers whose logarithm has been adjusted by α. That is, the value $F(x,y)$ behaves like a random integer of size $F(x,y) \cdot e^{\alpha(F)}$. So if $\alpha(F) < 0$ we consider F-values to be more likely to be smooth than random integers of the same size.

By inspection it is clear that $\alpha(F)$ receives its most negative contributions when q_p is large for very small p. That is, $\alpha(F)$ is more negative when F has many roots modulo small p.

3.2.4 Root Properties and d

It emerges from Sections 2.2.2 and 3.1 that asymptotically and in practice, the choice of $d = \deg F$ is crucial to controlling the size of the values inspected for smoothness during sieving. Now, root properties are determined by the distribution of the roots of F in $\mathbb{Z}_p$ for small p (that is, by the polynomial factorisation of F over $\mathbb{Z}_p$). Therefore, the choice of d will influence root properties as well. Quintic polynomials for example, can have more roots in $\mathbb{Z}_p$ for $p \geq 3$ than quadratic polynomials, but (on average) do they?

A thorough examination of this topic is not within the scope of this thesis, and in any event becomes less relevant in light of the procedures outlined in Chapter 5. In this subsection we intend only touching on some relevant and accessible considerations.

Eventually in this subsection we consider polynomials with $d = 4, 5$ and 6, as in Section 3.1. We use a simple model to estimate the average distribution of *non-projective* q_p for base-m polynomials of these degrees.

First though, we focus on $d = 2$ (that is, the polynomials produced by Montgomery's Two-Quadratics method, Section 2.3.1). The polynomials $F(x, y)$ are now binary quadratic forms. The rich theory of binary quadratic forms provides results from which we prove that, on average, the odds are stacked against F having good root properties. This highlights the importance of having regard to root properties in polynomial selection.

Quadratic Polynomials

A binary form is said to *represent* some $r \in \mathbb{Z}$ if there exist $x, y \in \mathbb{Z}$ for which $F(x, y) = r$. We are interested in the case $\gcd(x, y) = 1$.

Definition 3.2.2 *A binary form F primitively represents some $r \in \mathbb{Z}$ if there exist coprime integers x and y for which $F(x, y) = r$.*

The following theorem is a standard result from the theory of quadratic forms (see for example [13]). It gives necessary conditions on the primitive representation of integers by binary quadratic forms over $\mathbb{Z}$.

Theorem 3.2.3 *Let $F(x, y) = a_2x^2 + a_1xy + a_0y^2$ be a quadratic form over $\mathbb{Z}$ and Δ its discriminant. Then F primitively represents $r \in \mathbb{Z}$ only if there exists some $s \in \mathbb{Z}$ for which*

$$s^2 \equiv \Delta \bmod 4r. \tag{3.14}$$

As an immediate consequence of Theorem 3.2.3 we have

Corollary 3.2.4 *Let F be a binary quadratic form over $\mathbb{Z}$ with discriminant Δ, and let p be an odd prime not dividing Δ. If F primitively represents some $r \in \mathbb{Z}$ and $p|r$ then*

$$\left(\frac{\Delta}{p}\right) = 1.$$

In general the converse of Theorem 3.2.3 is not quite true, but there is a slightly more general statement that holds. If a solution to (3.14) exists then some class of forms of discriminant Δ primitively represents r (again, see [13]).

Using Corollary 3.2.4 we give a result leading to an estimate of the chance that $(\Delta/p) = 1$ for a random assignment p of the coefficients (a_2, a_1, a_0) of F.

Lemma 3.2.5 *For each odd prime p coprime to Δ, the number of non-trivial 3-tuples $(a_2, a_1, a_0) \bmod p$ for which $(\Delta/p) = 1$ is*

$$\frac{p}{2}(p^2 - 1).$$

Proof: Fix $a_1 \not\equiv 0 \bmod p$. For $\Delta = a_1^2 - 4a_2a_0$ and $(\Delta/p) = 1$ we have

$$a_2a_0 \equiv (-4)^{-1}(\chi_p - a_1^2) \bmod p, \tag{3.15}$$

where χ_p is any of the $(p-1)/2$ quadratic residues mod p. Hence the product a_2a_0 may take any of $(p-1)/2$ values mod p, exactly one of which will force the right hand side of (3.15) to be zero because exactly one $\chi_p = a_1^2$.

For each of the $(p-3)/2$ *non-zero* values of the right hand side of (3.15), there are $p-1$ ordered pairs (a_2, a_0) whose product gives the right hand side, since for each non-zero a_2, a_0 is uniquely determined by $a_0 = a_2^{-1}(-4)^{-1}(\chi_p - a_1^2) \bmod p$.

For each single zero value of the right hand side of (3.15), there are $2p-1$ ordered pairs (a_2, a_0) for which at least one of $a_2 \equiv 0 \bmod p$ or $a_0 \equiv 0 \bmod p$ holds.

Hence, for non-zero a_1, there are

$$\frac{p-3}{2}(p-1) + 2p - 1$$

ordered pairs (a_2, a_0) giving $(\Delta/p) = 1$. There are $p-1$ non-zero residue classes for a_1, so non-zero a_1 account for

$$(p-1)\left[\frac{p-3}{2}(p-1) + 2p - 1\right] \;=\; (p-1)\left[\frac{p^2}{2} + \frac{1}{2}\right]$$

tuples $(a_2, a_1, a_0) \bmod p$.

Now, if $a_1 \equiv 0 \bmod p$, we require

$$a_2a_0 \equiv (-4)^{-1}\chi_p \bmod p \tag{3.16}$$

where again χ_p is any of the $(p-1)/2$ quadratic residues mod p. Since the right hand side of (3.16) is always non-zero, there are $p-1$ pairs (a_2, a_0) for each χ_q, giving a total of

$$\frac{p-1}{2}(p-1)$$

tuples (a_2, a_1, a_0) mod p for $a_1 \equiv 0$ mod p.

So, the total number of tuples is

$$(p-1)\left[\frac{p^2}{2}+\frac{1}{2}\right]+\frac{p-1}{2}(p-1)=\frac{p}{2}\left[p^2-1\right].$$

∎

Thus, for odd p, the probability that a uniformly random non-trivial selection (a_2, a_1, a_0) mod p satisfies $(\Delta/p) = 1$ is given by

$$\begin{aligned}\operatorname{Prob}\left[(\Delta/p)=1\right] &= \frac{\frac{p}{2}(p^2-1)}{p^3-1}\\ &= \frac{1}{2}\left(1-\frac{p}{p^3-1}\right)\\ &< \frac{1}{2}\left(1-\frac{1}{p^2}\right).\end{aligned}$$

For odd p not dividing Δ, it is therefore more likely than not that $(\Delta/p) = -1$, and the probability that $(\Delta/p) = 1$ is smallest for smaller p. This highlights the significance of selecting polynomials which do have roots modulo small p.

Higher Degree Polynomials

We turn now to polynomials of higher degree. Recall from Section 3.1 that base-m polynomials of degree $4, 5$ and 6 are the most relevant for integers in the current range of interest. Unfortunately, when passing from $d = 2$ to higher degree, we lose the rigour of available results on quadratic forms. Instead, we now obtain information about the factorisation of the single variable polynomial $f(x)$ mod p as a function of d (assuming $p > d$), from a result concerning the Galois group of a random polynomial $f \in \mathbb{Z}[x]$.

Informally, the result is that most monic polynomials $f \in \mathbb{Z}[x]$ of degree d have Galois group isomorphic to the symmetric group on d elements, S_d. That being the case, the typical factorisation of f can be deduced by examining the space of possible cycle decompositions in S_d.

Formally, the result is as follows. Let $f(x) \in \mathbb{Z}[x]$ of degree d be monic. The Galois group of f, $G(f)$, may be considered as a subgroup of S_d. The question is this: how many f have $G(f) < S_d$?

Theorem 3.2.6 *Let $E_d(A)$ be the number of monic polynomials $f(x) \in \mathbb{Z}[x]$ with $\max(|a_0|, \ldots, |a_d|) \leq A$ for which $G(f) < S_d$. Then*

$$E_d(A) \ll A^{d-1/2} \log^{1-\epsilon} A$$

where $\epsilon = \epsilon(d) > 0$.

For a proof see [34]. For more discussion on this and related results also see [19] and [24]. We typically have $d = 5$ and $A = 10^{24}$ with, notionally, $0 \leq a_i < A$. The total number of possible monic polynomials is A^5. Theorem 3.2.6 gives

$$\frac{E_d(A)}{A^5} \ll 5.53 \cdot 10^{-11}.$$

Hence, we may safely assume that most monic polynomials we encounter have

$$G(f) \cong S_d.$$

However, we search only amongst non-monic polynomials. The roots mod p of F that arise from $a_d > 1$ are precisely the projective roots. As we see in Chapters 5 and 6, projective roots are exploited to equip each F with better than average root properties. The parameter $\alpha(F)$ incorporates this effect. For now, we use Theorem 3.2.6 as a guide only to the underlying non-projective root structure for each F.

We now examine this structure, on the assumption that $G(f) \cong S_d$. A permutation $\sigma \in S_d$ which is a product of k cycles of length $l_1, \ldots, l_k$ (of course $\sum_{i=1}^{k} l_i = d$) corresponds to a factorisation of f into k irreducible factors of respective degrees l_i for $i = 1, \ldots, k$. This assumes that each cycle is represented to its maximal length - that is, we do not for example break a 3-cycle into two transpositions. We refer to each possible set of l_i with $\sum_{i=1}^{k} l_i = d$ as a distinct *cycle structure* in S_d. The exercise now is to count the occurrences of each cycle structure for $d = 4, 5, 6$.

Table 3.4 shows the number of ways each possible cycle structure appears in S_4, S_5 and S_6.

Each appearance of a cycle of length one ($l_i = 1$) in a given cycle structure corresponds to a distinct root mod p of f. Hence, Table 3.5 collects for each d, the structures that give $0, \ldots, d$ roots of f. The frequency column records the frequency with which structures giving q_p roots mod p occur as a fraction of the $d!$ possibilities.

Notice that $q_p > 1$ on average 29% of the time when $d = 4$, 26% of the time when $d = 5$ and 27% of the time when $d = 6$. This indicates that the average set of non-projective root properties is best for $d = 4$ and worst for $d = 5$, although the difference between them is not great.

In any event, the procedure we present in Chapter 5 isolates polynomials with exceptionally good root properties, not polynomials with average root properties. Whilst experimenting with different degrees, we observed that the best values of $\alpha(F)$ we found do not vary much across $d = 4, 5, 6$. Hence we conclude that in practice, the choice of degree should be determined mainly by size considerations, not by root properties.

3.3 Summary

In this chapter we have identified and described the properties which influence yield.

The influence of size on yield is, for the most part, well known. However, the problem of choosing the degrees of (F_1, F_2) is unique to the number field sieve. Here we have verified that $d = 4, 5, 6$ are the relevant degrees for non-monic base-m polynomials and N in the range of interest. For most of this range, $d = 5$ is the best degree for F_1.

To assess the influence of root properties on yield it is necessary to derive a parameter which quantifies their effect. We use a "typical F-value" model, the crux of which is to estimate accurately the quantity $\text{cont}_p(v)$ for certain values v. After estimating $\text{cont}_p(v)$ we have constructed a parameter $\alpha(F)$ which measures root properties in the following sense: due to root properties, F-values $F(x, y)$ behave as if they are random values of size $F(x, y) \cdot e^{\alpha(F)}$. The idea now is to seek polynomials with $\alpha(F) \ll 0$.

We have also considered the choice of d as an influence on root properties. We find that, on average and over the non-projective roots, $d = 4$ is better than $d = 6$ and $d = 5$, in that order. However, the difference is not so great that these considerations should enter into the choice of d.

It is instructive at this point to hint at the benefit obtained from understanding root properties. Using the procedures of Chapter 5, it is not uncommon to find non-monic quintic F_1 (with common root, say, m_1) for N with $\alpha(F_1) \approx -7$. Indeed, that is the case with the polynomial used for the RSA-140 factorisation. How much benefit does this return?

Since $e^7 \approx 1000$, values of such F_1 behave as if they are $1/1000$ their actual value. Suppose we attempted to reap the same reward, naively, by shaving a factor of 1000 from each coefficient of F_1. Then we would have a polynomial whose coefficients are of the size expected for a random choice of polynomial with $m_2 \approx m_1/1000$. Since $m \approx N^{1/(d+1)}$ the new polynomial has coefficients of the size expected from a random choice for

$$N_2 \approx m_2^{d+1} = \frac{N_1}{1000^{d+1}}.$$

That is, $N_2 \approx 10^{-18} N_1$. The benefit from root properties alone, once quantified, is that the polynomials we find have yields expected from a random choice of polynomial for an integer 19 digits smaller than the integer we are trying to factorise.

This is the influence of root properties alone. How does size interact with root properties? Once we have a means of quantifying root properties, we are drawn to questions of their interaction with size, and the influence of both properties on yield. These are typical of the questions we consider in the next chapter.

S_d	$\{l_i\}$	Occurences
S_4	1,1,1,1	1
	1,1,2	6
	1,3	8
	2,2	3
	4	6
		24
S_5	1,1,1,1,1	1
	1,1,1,2	10
	1,1,3	20
	1,4	15
	2,2,1	30
	2,3	20
	5	24
		120
S_6	1,1,1,1,1,1	1
	1,1,1,1,2	15
	1,1,1,3	40
	1,1,4	90
	1,1,2,2	45
	1,5	144
	1,2,3	120
	2,2,2	15
	2,4	90
	3,3	40
	6	120
		720

Table 3.4: Cycle structure counts in S_4, S_5 and S_6

$d = 4$		$d = 5$		$d = 6$	
q_p	Freq.	q_p	Freq.	q_p	Freq.
0	0.38	0	0.37	0	0.37
1	0.33	1	0.38	1	0.37
2	0.25	2	0.17	2	0.19
3	0.00	3	0.08	3	0.06
4	0.04	4	0.00	4	0.02
		5	0.01	5	0.00
				6	0.00

Table 3.5: Relative frequencies of q_p as a function of d.

Chapter 4

Modelling Yield

This chapter is a computational study on polynomial yield. The aims are to ensure that our understanding of the properties which influence yield is correct, to extract some information on the benefit obtained from manipulating these properties, and to present a simple method for estimating yield.

For the most part we study only simple cases in this chapter. In particular, we consider polynomials selected by Montgomery's Two Quadratics method (see Section 2.3.1), which have been adjusted for line sieving across $F(x, 1) = f(x)$. Line sieving over quadratic polynomials is certainly of interest in its own right (see [33] and [32]), and has the added advantages of being easier to visualise and tidier to analyse. Hence, unless stated otherwise, we assume in this chapter that $F(x, y)$ is of degree two and has been chosen for sieving across $f(x)$.

Recall from the previous chapter that the parameter $\alpha(f)$ is constructed to quantify the effect on the typical f-value of root properties. Indeed, we take the value $f(x)$ to behave like a random integer of size $f(x) \cdot e^{\alpha(f)}$. In Section 4.1 we ask "how much benefit can be obtained from exploiting root properties?". That is, we examine yield as a function of α, with α in an achievable range.

At this stage we use an established method of calculating yield. We adapt the method used by Boender in [6] to calculate yield of MPQS polynomials. Boender confirms in [6] that his method gives a reasonable approximation to the yield of such polynomials. We also adapt Boender's method to consider 1LP- and 2LP- yields as a function of α. Finally, we conduct sieving experiments to confirm the predictions of this section.

In Section 4.2 we use the fact that α correctly quantifies the effect of root properties to suggest a simple method of approximating yield. We test the estimate on several polynomials from [33]. We compute peak yields of these polynomials, and yield across the sieve region, then compare predicted yields with actual yields found by sieving. We then use our simpler model to examine yield due to root properties under conditions that we encounter whilst considering larger N and higher degree polynomials in subsequent chapters.

Section 4.3 contains a summary of this chapter.

4.1 Yield as a Function of Root Properties

We turn now to the influence of root properties on yield. Below we adapt Boender's method of calculating yield to our polynomials, and then compute variations in full, 1LP- and 2LP- yields as a function of α. That is, we ask "all other things being equal, what is the influence of root properties on yield?".

4.1.1 In Theory: Boender's Yield Estimation

Boender's approach is to use analytic estimates of the number of smooth integers in short intervals. Taking f to be a continuous curve on $\mathbb{R}$, these estimates are computed on intervals in $\mathbb{R}$ sufficiently small to approximate the likelihood of a given point in that interval being an integer point on f. Estimates are then summed over many intervals.

Smooth Integers in an Interval

We require an estimate of the number smooth integers in an interval of a given size. For an integer n recall that $P_1(n)$ denotes the largest prime factor of n and that

$$\psi(x,y) = \left|\left\{n \in \mathbb{Z}^+ : n \leq x \text{ and } P_1(n) \leq y\right\}\right|.$$

Then asymptotically

$$\psi(x,y) \approx x\left(\rho(u) + (1-\gamma)\frac{\rho(u-1)}{\log x}\right) \tag{4.1}$$

where $u = (\log x)/\log y$, γ is Euler's constant and $\rho(u)$ is the Dickman function (see Section 2.2.1).

Now, for fixed $\epsilon \in (0,1)$, the number of y-smooth integers in the interval $[x, x+x/z]$ is given by

$$\psi\left(x+\frac{x}{z},y\right) - \psi(x,y) = \frac{\log(1+y/\log x)}{z\log y}\,\psi(x,y)\left[1+O_\epsilon\left(\frac{1}{z}+\frac{\log\log(1+y)}{\log y}\right)\right] \tag{4.2}$$

for x, y, z in the range $x \geq 2$ and

$$(\log\log x)^{2/3+\epsilon} < \log y \leq (\log x)^{2/5},\ 1 \leq z \leq R(x,y),$$

where $R(x,y)$ is an expression depending on x, y and some fixed constants (see [6]).

Combining (4.1) and (4.2) and approximating some of the logarithms gives

$$\frac{z}{x}\left\{\psi\left(x+\frac{x}{z},y\right) - \psi(x,y)\right\} \approx \left(1-\frac{\log\log x}{\log y}\right)\sigma(x,y)\left(1+c_1(\epsilon)\frac{1}{z}+c_2(\epsilon)\frac{\log\log y}{\log y}\right), \tag{4.3}$$

where $\sigma(x, y)$ is given by

$$\sigma(x, y) = \rho(u) + (1 - \gamma)\frac{\rho(u-1)}{\log x}$$

and the $c_i(\epsilon)$ are constants depending on ϵ [6]. Boender notes that the range of interest for x, y, z slightly extends that for which (4.2) is proven to hold. Empirically however, (4.2) still provides a good approximation in the range of interest.

Estimating Yield

Suppose we are to sieve for B-smooth values of $|f(x)|$ with x in the range $[a_1, a_2]$. We use the approximations at (4.3) to estimate the yield of f across $[a_1, a_2]$.

Care is required in the calculations below when f (considered as a continuous curve on the real interval $[a_1, a_2]$) contains a stationary point or real roots in $[a_1, a_2]$. In our circumstances this is always the case. Clearly each curve f can be cut into segments which exclude roots and turning points (we require at most four segments). We call the segment so obtained which occupies the largest portion of $[a_1, a_2]$ the *principal segment*. For each curve below we have repeated our calculations on every segment of the curve, and obtained almost identical results on each segment. Hence we report only the results on the principal segment.

Let I be the real x-interval defining the principal segment, and let Γ be the continous curve defined by f on I. Since Γ contains no turning point in I, we can assume either $f'(x) < 0$ or $f'(x) > 0$ for all $x \in I$. We assume the latter, the former only requires sign changes in the arguments below. Similarly, we assume $f(x) > 0$ for all $x \in I$. The question now is 'how many integer points on Γ are B-smooth?'.

We approximate the number of B-smooth integer values on Γ by cutting Γ into shorter intervals and summing the yield over these intervals. Let S_1 and S_2 be the minimum and maximum values respectively, taken by Γ on I. Cut $[S_1, S_2]$ into K subintervals $[y_i, y_{i+1}]$ for $i = 0, \ldots, K-1$ by taking

$$h = \frac{\log S_2 - \log S_1}{K},$$

so $y_i = S_1 e^{ih}$. In accordance with our notation for estimating the number of smooth integers in an interval, we write $y_{i+1} = y_i + y_i/z$ where $1/z = e^h - 1$.

Now, for each y_i, let $x_i \in \mathbb{R}$ be such that $(x_i, y_i) \in \Gamma$. Let

$$s_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

denote the slope of Γ on $[x_i, x_{i+1}]$, and let $t(y_i)$ denote the number of B-smooth f-values on Γ with $y \leq y_i$. Clearly the yield on the whole of Γ, X_f, is given by

$$X_f = \sum_{i=0}^{K-1} \left(t(y_{i+1}) - t(y_i)\right). \tag{4.4}$$

For $y \in [y_i, y_{i+1}]$ the probability that a randomly chosen $(x, y) \in \Gamma$ has $x \in \mathbb{Z}$ is approximately $1/s_i$. So we have

$$t(y_{i+1}) - t(y_i) \approx \frac{y_{i+1} - y_i}{s_i} P(f_i, B) = (x_{i+1} - x_i) P(f_i, B)$$

where $P(f_i, B)$ is the probability that an integer f-value in $[y_i, y_{i+1}]$ is B-smooth.

Recall that we consider f-values $f(x)$ as likely to be B-smooth as random integers of logarithm $\log(f(x)) + \alpha(f)$ where

$$\alpha(f) = \sum_{p \le B} \left(\frac{1}{p-1} - \mathrm{cont}_p(f) \right) \log p.$$

So, if $g_i(\alpha) = \log y_i + \alpha = \log S_1 + ih + \alpha$, and if $v_i(\alpha) = g_i(\alpha)/\log B$, approximation (4.3) yields

$$\begin{aligned} t(y_{i+1}) - t(y_i) &\approx \\ &(x_{i+1} - x_i) \left(1 - \frac{\log g_i(\alpha)}{\log B}\right) \left(\rho(v_i(\alpha)) + (1-\gamma)\frac{\rho(v_i(\alpha) - 1)}{g_i(\alpha)}\right) \\ &\times \left(1 + \frac{c_1}{z} + \frac{c_2 \log\log B}{\log B}\right). \end{aligned} \tag{4.5}$$

Approximation (4.5) and equation (4.4) give an approximation to X_f.

4.1.2 Full Yield as a Function of α

We now consider the full yield X_f as a function of α. For B fixed, $\alpha(f)$ is bounded. In fact for $B = 5 \cdot 10^6$ with quadratic f we have approximately $|\alpha| \le 14.16$. However for the quadratic polynomials investigated, typically $\alpha \in [-3, 1]$, a range of 4. So we consider $\alpha \in [-4, 0]$ and refer to this as the *practical range* for α. Note that when we consider higher degree polynomials for larger N we encounter much more extreme values of α.

We approximate X_f, with appropriate parameter choices, as α varies in the practical range, all other things being equal. In fact we calculate

$$\begin{aligned} Q(\alpha) &= \frac{X_f(\alpha)}{X_f(0)} \\ &\approx \frac{\sum_{i=0}^{K-1} (x_{i+1} - x_i) \left(1 - \frac{\log g_i(\alpha)}{\log B}\right) \left(\rho(v_i(\alpha)) + (1-\gamma)\frac{\rho(v_i(\alpha)-1)}{g_i(\alpha)}\right)}{\sum_{i=0}^{K-1} (x_{i+1} - x_i) \left(1 - \frac{\log g_i(0)}{\log B}\right) \left(\rho(v_i(0)) + (1-\gamma)\frac{\rho(v_i(0)-1)}{g_i(0)}\right)}. \end{aligned}$$

The quantity $Q(\alpha)$ approximates the relative increase in full yield we might expect as α decreases in the practical range.

Note 4.1.1 In practice $Q(\alpha)$ is approximately independent of K, so we use $K = 100$ in accordance with [6].

We now insert typical polynomials and other parameters into the calculations. We use polynomials selected for factorisations of five integers C87, C97, C105, C106 and C107 in [33]. The polynomials used are the polynomials labelled $f_1(x)$ in [33] for each integer. Other relevant parameters from [33] are shown in Table 4.1.

	C87	C97	C105	C106	C107
divides	$72^{99}+1$	$12^{441}+1$	$3^{367}-1$	$12^{157}+1$	$6^{223}+1$
B	$1.0 \cdot 10^6$	$2.2 \cdot 10^6$	$1.6 \cdot 10^6$	$2.7 \cdot 10^6$	$2.9 \cdot 10^6$
$\|x\| \leq$	$7.5 \cdot 10^{12}$	$25 \cdot 10^{12}$	$7.5 \cdot 10^{14}$	$1.0 \cdot 10^{15}$	$1.0 \cdot 10^{15}$
$I \approx$	$[3.0 \cdot 10^{12}, 7.5 \cdot 10^{12}]$	$[-2.5 \cdot 10^{13}, -5.8 \cdot 10^{12}]$	$[-1.3 \cdot 10^{13}, 7.5 \cdot 10^{14}]$	$[-1.0 \cdot 10^{15}, -2.2 \cdot 10^{14}]$	$[-1.3 \cdot 10^{14}, 1.0 \cdot 10^{15}]$

Table 4.1: Parameters for Table 4.2

Values of $Q(\alpha)$ for these parameters approximate the range of relative yields we can expect due to root properties on typical polynomials in the above cases. Table 4.2 contains $Q(\alpha)$ calculated at several α.

	$Q(\alpha)$				
$-\alpha$	$C87$	$C97$	$C105$	$C106$	$C107$
0.05	1.01	1.01	1.01	1.01	1.01
0.50	1.11	1.11	1.12	1.11	1.11
1.00	1.24	1.22	1.24	1.23	1.23
1.50	1.37	1.35	1.39	1.37	1.36
2.00	1.53	1.50	1.54	1.51	1.51
2.50	1.69	1.66	1.72	1.68	1.68
3.00	1.88	1.83	1.92	1.86	1.86
3.50	2.09	2.02	2.13	2.06	2.06
4.00	2.32	2.23	2.38	2.28	2.28

Table 4.2: $Q(\alpha)$ vs α

The complete results on C107 for $\alpha \in [-4, 0]$ are shown in Figure 4.1 below. The complete results for the other parameters are similar.

We see that, heuristically, we expect the difference in yield between polynomials with values of α at the extremes of the practical range to be as much as a factor of two. This is a significant difference.

4.1.3 1LP-Yield as a Function of α

Suppose we now have B_1 and B_2, with $B_1 < B_2$, and consider the f-values that are B_1 smooth but for the appearance of exactly one prime between B_1 and B_2. Let Y_f be the

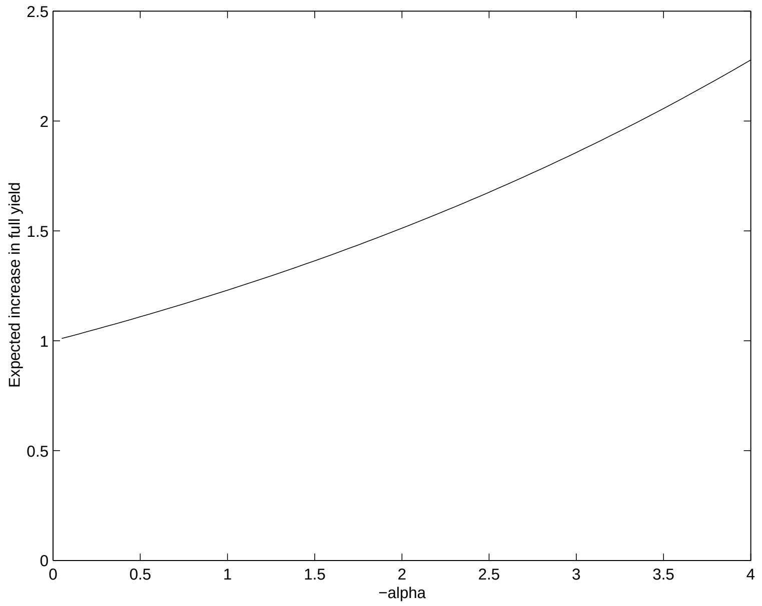

Figure 4.1: $Q(\alpha)$ for C107

number of these 1LP-smooth f values on Γ. Again, we approximate Y_f by examining the 1LP-yield in intervals along Γ. Let $t_1(y_i)$ be the number of 1LP-smooth f-values on Γ with $y \leq y_i$. Clearly

$$Y_f = \sum_{i=0}^{K-1} (t_1(y_{i+1}) - t_1(y_i)). \tag{4.6}$$

In what follows we implicitly assume that if $P_1(n)$ is the largest prime factor of some integer n, then the prime factors of $n/P_1(n)$ are distributed like those of a random integer of size $n/P_1(n)$. In fact this is not true - see Section 2.2.1 and Aside 4.2.2 following. However, the assumption suffices for our purposes.

For each large prime p, let $g_{i,p}(\alpha) = g_i(\alpha) - \log p = \log S_1 + ih + \alpha - \log p$, and $v_{i,p}(\alpha) = g_{i,p}(\alpha)/\log B$. Then

$$\begin{aligned} t_1(y_{i+1}) - t_1(y_i) \approx & \sum_{B_1<p<B_2} \frac{q_p}{p} \left(t(y_{i+1}/p) - t(y_i/p)\right) \\ \approx \quad & (x_{i+1} - x_i) \sum_{B_1<p<B_2} \frac{q_p}{p} \left(1 - \frac{\log g_{i,p}(\alpha)}{\log B_1}\right) \left(\rho(v_{i,p}(\alpha)) + (1-\gamma)\frac{\rho(v_{i,p}(\alpha)-1)}{g_{i,p}(\alpha)}\right) \\ & \times \left(1 + \frac{c_1}{z} + \frac{c_2 \log\log B_1}{\log B_1}\right), \end{aligned} \tag{4.7}$$

which, with (4.6) gives an approximation to Y_f.

We are interested in the relative increase in 1LP-yield as a function of α, that is, the ratio

$$\frac{Y_f(\alpha)}{Y_f(0)}. \tag{4.8}$$

Calculating (4.8) directly is time-consuming. Since we are interested only in checking that practical changes in α can bring significant increases in yield, we instead obtain upper and lower bounds on (4.8) in intervals along f. The bounds suffice to show a significant increase in yield. For $i = 1, \ldots, K-1$ let $Y_{f,i}(\alpha) = t_1(y_{i+1}) - t_1(y_i)$ be the partial yield of f in the i-th interval only. We bound

$$R_i(\alpha) = \frac{Y_{f,i}(\alpha)}{Y_{f,i}(0)}$$

for $i = 1 \ldots K-1$.

Recall that Δ denotes the discriminant of f. Let

$$LP = \{p : p \text{ prime },\ B_1 < p < B_2,\ (\Delta/p) = 1\}$$

be the set of large primes which may appear in the factor base and let p_1, p_2 be the minimum and maximum elements (respectively) in LP. Then

$$\begin{aligned} g_{i,1}(\alpha) &= \log x_i + ih + \alpha - \log p_2, \text{ and} \\ g_{i,2}(\alpha) &= \log x_i + ih + \alpha - \log p_1 \end{aligned}$$

are the minimum and maximum values (respectively) of $g_{i,p}(\alpha)$ on (x_i, x_{i+1}). Also,

$$\begin{aligned} v_{i,1}(\alpha) &= g_{i,1}(\alpha)/\log B_1, \text{ and} \\ v_{i,2}(\alpha) &= g_{i,2}(\alpha)/\log B_1 \end{aligned}$$

are the minimum and maximum values (respectively) of $v_{i,p}$ on (x_i, x_{i+1}). Finally, let

$$\begin{aligned} \mathcal{L}_i(\alpha) &= \frac{2}{p_2}\left(1 - \frac{\log g_{i,2}(\alpha)}{\log B}\right)\left(\rho(v_{i,2}(\alpha)) + (1-\gamma)\frac{\rho(v_{i,2}(\alpha)-1)}{g_{i,2}(\alpha)}\right), \\ \mathcal{U}_i(\alpha) &= \frac{2}{p_1}\left(1 - \frac{\log g_{i,1}(\alpha)}{\log B}\right)\left(\rho(v_{i,1}(\alpha)) + (1-\gamma)\frac{\rho(v_{i,1}(\alpha)-1)}{g_{i,1}(\alpha)}\right). \end{aligned}$$

Then

$$Y_{f,i}(\alpha) < (x_{i+1} - x_i) \cdot |LP| \cdot \mathcal{U}_i(\alpha).$$

Similarly, $Y_{f,i}(\alpha) > (x_{i+1} - x_i) \cdot |LP| \cdot \mathcal{L}_i(\alpha)$. Since we are varying only α,

$$\frac{\mathcal{L}_i(\alpha)}{\mathcal{U}_i(0)} \ < \ R_i(\alpha) < \ \frac{\mathcal{U}_i(\alpha)}{\mathcal{L}_i(0)}. \tag{4.9}$$

To calculate $R_i(\alpha)$ we use the additional parameters from [33] given in Table 4.3.

	C87	C97	C105	C106	C107
B_1	$1.0 \cdot 10^6$	$10 \cdot 10^6$	$23 \cdot 10^6$	$27 \cdot 10^6$	$27.2 \cdot 10^6$
B_2	$2.346 \cdot 10^6$	$24 \cdot 10^6$	$30 \cdot 10^6$	$30 \cdot 10^6$	$30 \cdot 10^7$

Table 4.3: Large prime bounds

$\frac{\mathcal{L}_i(-4)}{\mathcal{U}_i(0)}, \frac{\mathcal{U}_i(-4)}{\mathcal{L}_i(0)}$	C87	C97	C105	C106	C107
$i = 1$	$0.64, 4.42$	$0.61, 4.39$	$1.52, 1.93$	$1.51, 1.93$	$1.54, 1.92$
$i = 25$	$0.69, 4.98$	$0.65, 4.82$	$1.62, 2.05$	$1, 62, 2.06$	$1.62, 2.05$
$i = 50$	$0.72, 5.36$	$0.68, 5.18$	$1.72, 2.18$	$1.72, 2.20$	$1.72, 2.18$
$i = 75$	$0.75, 5.74$	$0.71, 5.53$	$1.81, 2.29$	$1.83, 2.32$	$1.81, 2.30$
$i = 99$	$0.78, 5.97$	$0.73, 5.85$	$1.87, 2.40$	$1.88, 2.42$	$1.90, 2.40$

Table 4.4: Upper and lower bounds on $R_i(-4)$

We give values of the bounds on $R_i(\alpha)$ evaluated at $\alpha = -4$, for several i, in Table 4.4.

The values for C87 and C97 are inconclusive, our bounds on the large primes appearing here are too crude for integers of this size. But the values for C105, C106 and C107 (in particular the lower bounds) are useful. We illustrate in Figure 4.2 the complete results for C107. The results for C105 and C106 are similar. The region between the lines represents the expected increase in the 1LP-yield of f.

We conclude that practical changes in α can also bring significant increases in 1LP-yield.

4.1.4 2LP-Yield as a Function of α

Let Z_f be the number of 2LP-smooth f values on Γ. Let $t_2(y_i)$ be the number of 2LP-smooth f-values on Γ with $y \leq y_i$. Then

$$Z_f = \sum_{i=0}^{K-1} (t_2(y_{i+1}) - t_2(y_i)). \tag{4.10}$$

For the large prime pair $\{p, q\}$ let $g_{i,pq}(\alpha) = g_i(\alpha) - \log p - \log q$ and $v_{i,pq}(\alpha) = g_{i,pq}(\alpha)/\log B_1$. Then, assuming again (which is not quite true) that the appearance

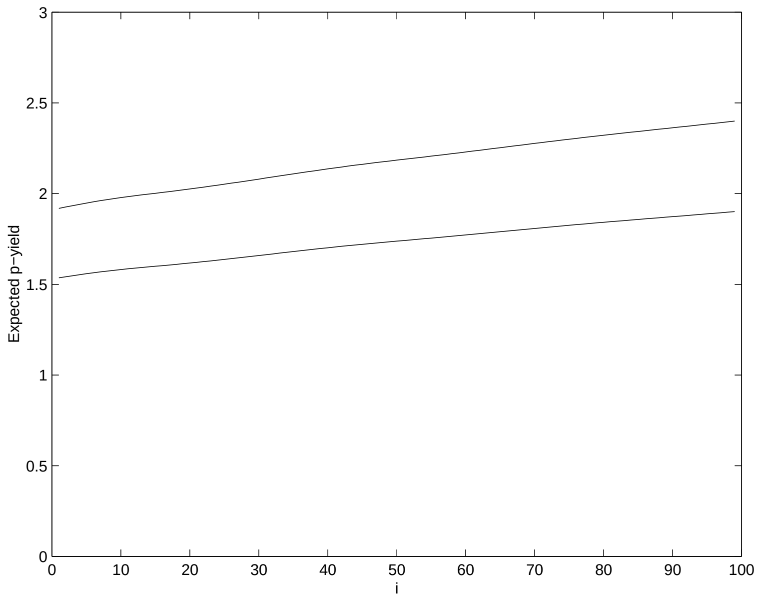

Figure 4.2: $R_i(-4)$ for C107

of p and q in the factorisations of f-values is independent,

$$
\begin{aligned}
t_2(y_{i+1}) - t_2(y_i) \approx & \sum_{\{p,q\}\in LP} \frac{4}{pq}\left(t(y_{i+1}/pq) - t(y_i/pq)\right) \\
\approx \quad & (x_{i+1}-x_i) \sum_{\{p,q\}\in LP} \frac{4}{pq}\left(1-\frac{\log g_{i,pq}(\alpha)}{\log B_1}\right)\left(\rho(v_{i,pq}(\alpha)) + (1-\gamma)\frac{\rho(v_{i,pq}(\alpha)-1)}{g_{i,pq}(\alpha)}\right) \\
& \times \left(1+\frac{c_1}{z}+\frac{c_2 \log\log B_1}{\log B_1}\right). \qquad (4.11)
\end{aligned}
$$

Equation (4.10) and approximation (4.11) give an approximation to Z_f.

Again we present bounds on the relative increase in Z_f in intervals along Γ, as α varies in the practical range. Let $Z_{f,i} = t_2(y_{i+1}) - t_2(y_i)$ be the 2LP-yield of f in the i-th interval, and let

$$T_i(\alpha) = \frac{Z_{f,i}(\alpha)}{Z_{f,i}(0)}.$$

We calculate bounds on T_i for $i = 1, \ldots, K-1$ by repeating the calculations of the previous section. Thus, let p_1, p_2 and p_3, p_4 be the two least and two greatest elements (respectively) of LP. Let

$$
\begin{aligned}
g_{i,1}(\alpha) &= \log x_i + \alpha - \log p_3 - \log p_4, \\
g_{i,2}(\alpha) &= \log x_i + \alpha - \log p_1 - \log p_2, \\
v_{i,1}(\alpha) &= g_{i,1}(\alpha)/\log B_1, \text{ and} \\
v_{i,2}(\alpha) &= g_{i,2}(\alpha)/\log B_1.
\end{aligned}
$$

Then if

$$\mathcal{L}_i(\alpha) = \frac{4}{p_3p_4}\left(1-\frac{\log g_{i,2}(\alpha)}{\log B}\right)\left(\rho(v_{i,2}(\alpha)) + (1-\gamma)\frac{\rho(v_{i,2}(\alpha)-1)}{g_{i,2}(\alpha)}\right), \text{ and}$$
$$\mathcal{U}_i(\alpha) = \frac{4}{p_1p_2}\left(1-\frac{\log g_{i,1}(\alpha)}{\log B}\right)\left(\rho(v_{i,1}(\alpha)) + (1-\gamma)\frac{\rho(v_{i,1}(\alpha)-1)}{g_{i,1}(\alpha)}\right)$$

we have

$$\frac{\mathcal{L}_i(\alpha)}{\mathcal{U}_i(0)} < T_i(\alpha) < \frac{\mathcal{U}_i(\alpha)}{\mathcal{L}_i(0)}. \tag{4.12}$$

Table 4.5 contains values of the bounds on $T_i(-4)$ given by (4.12), for several i.

$\frac{\mathcal{L}_i(-4)}{\mathcal{U}_i(0)}, \frac{\mathcal{U}_i(-4)}{\mathcal{L}_i(0)}$	C87	C97	C105	C106	C107
$i = 1$	$0.25, 11.60$	$0.23, 11.53$	$1.26, 2.02$	$1.26, 2.02$	$1.33, 2.07$
$i = 25$	$0.25, 12.32$	$0.23, 12.04$	$1.33, 2.12$	$1.33, 2.13$	$1.40, 2.19$
$i = 50$	$0.26, 13.13$	$0.24, 13.00$	$1.40, 2.26$	$1.41, 2.27$	$1.45, 2.28$
$i = 75$	$0.26, 13.68$	$0.24, 13.64$	$1.46, 2.36$	$1.47, 2.38$	$1.51, 2.38$
$i = 99$	$0.27, 14.36$	$0.25, 14.24$	$1.51, 2.47$	$1.53, 2.50$	$1.56, 2.46$

Table 4.5: Upper and lower bounds on $T_i(-4)$

The results for C87 and C97 are again inconclusive, whilst those for C105, C106 and C107 are useful. We conclude again that practical changes in α can bring significant increases in the 2LP-yield.

4.1.5 In Practice: Sieving Experiments

We now seek empirical verification that the parameter $\alpha(f)$ indeed captures the effect of root properties on yield.

Differences in yield amongst polynomials f_1 and f_2, due only to root properties, can be observed by examining the yield across regions where $f_1 \approx f_2$. We chose five candidate polynomials, Polynomials $A, B, \ldots, E$, for the 106 digit integer C106 given in Table 4.1. The polynomials are given in Appendix A. These particular polynomials were chosen because they exhibit a certain range of root properties. We sieved each polynomial $B, \ldots, E$ in intervals of size 10^8 centred on a point at which the polynomials take the same value as Polynomial A. Over the entire interval the "other" polynomial has the same size as Polynomial A to at least the fourth significant figure, and usually more. Any difference in yield between the polynomials over these intervals should therefore be due their different root properties.

These polynomials are typical of polynomials produced by Montgomery's Two Quadratics method for line sieving on C106, except that their root properties are fixed. In fact, Polynomials $A, \dots, E$ have $\alpha \in [-2.56, 1.51]$ as shown Table 4.6. We used $B_1 = 2700000$ and $B_2 = 30000000$ in accordance with [33].

f	$\alpha(f)$
A	-2.56
B	-1.50
C	-0.50
D	0.52
E	1.51

Table 4.6: α values for candidate polynomials.

We summarize the results in Table 4.7 below. The relative yields shown are the yield of Polynomial A relative to the "other" polynomial, so for example the full yield of Polynomial A is 2.32 times that of Polynomial E.

Polynomial f	$\alpha(f)-\alpha(A)$	rel. total yield	rel. full yield	rel. 1LP yield	rel. 2LP yield
B	1.06	1.46	1.55	1.54	1.39
C	2.06	1.92	2.09	1.99	1.83
D	3.08	1.94	2.20	1.99	1.84
E	4.07	2.03	2.32	2.08	1.95

Table 4.7: Relative yields due to root properties

According to the calculations of Section 4.1.2 the increases in *full* yield of A should be approximately $1.24, 1.51, 1.86, 2.30$ relative to polynomials $B, \dots, E$ respectively. Moreover, the increases in 1LP and 2LP yields of Polynomial A relative to Polynomial E fall close to the middle of the bounds of Sections 4.1.3 and 4.1.4.

The values taken by Polynomials C and D behave more like random integers than we expect on the basis of Section 4.1.2. Probably this is because in Section 4.1.2 we consider only changes in α, not the value itself. The values $\alpha(C)$ and $\alpha(D)$ are close to zero (-0.50 and 0.50 respectively). Hence we must expect their values to behave more like random integers than if their α values were -2 and -1 for example.

We conclude that in the quadratic cases examined, differences in yield from root properties alone can indeed be as much as a factor of two. Root properties are therefore a factor which should be considered whilst modelling yield.

4.2 Modelling Yield

Having established that $\alpha(f)$ seems to quantify well the effect of root properties, we now seek a simple model of yield. The model given here is used in subsequent chapters dealing with the problem of finding good polynomials.

In Section 4.2.1 below we visualise the relevant features of yield. From sieving experiments we see increased yields at real roots of f, and the relative variation in yield away from real roots. We refer to the former as *peak yield* and the latter as *yield across the region*. Notice that whilst the peak yields we see in this section all correspond to real roots, in general the peak yield of a polynomial occurs where it takes minimal absolute value. We propose a simple method of estimating yield in Section 4.2.2, and use it in Section 4.2.3 to predict both peak yield and yield across the region. To this point we have considered only quadratic polynomials with line sieving, but in Section 4.2.4 we extend our simple model to repeat some calculations of Section 4.1 under conditions experienced in factorisations of large RSA keys.

4.2.1 Actual Yield

On each of the polynomials $A, \dots, E$ we performed line sieving in short intervals along $|x| \leq 10^{15}$, again with smoothness bounds $B_1 = 27000000$ and $B_2 = 30000000$. We sieved in intervals of length 10^8 centred at steps of 10^{14} along the sieve interval, and in intervals of 10^8 centred at each real root of each polynomial.

For all polynomials the obvious feature of yield across the sieve region is the relative increase at real roots. This of course is due to the polynomials taking much smaller values close to roots. Common to all polynomials under the conditions we investigated is an increase in total yield by a factor of at least fifteen across roots. Polynomial A is typical, Figure 4.3 shows the relative increase at real roots of Polynomial A.

During an entire sieve run, values of x close to real roots of $f(x)$ are a richer supply of smooth f-values than those not. Of course this does not necessarily mean that we should blindly search for polynomials with as many real roots as possible. We see particularly in subsequent chapters that, leaving aside the question of root properties, the pervading requirement is that f-values be kept small over the sieve region. Real roots will help of course, but are not the sole determing factor.

Most values of x in the sieve region are not close to real roots of f. The total yield away from real roots is not quite as flat as Figure 4.3 indicates. Figure 4.4 shows total yield across $|x| \leq 10^{15}$ just in steps of 10^{14} (that is, without explicitly showing the yield at real roots).

Remark 4.2.1 Figure 4.4 suggests that, in relative terms, the yield of f varies greatly across the region. This has consequences for the collection of relations. Recall that a relation for the number field sieve in its full generality is a coprime integer pair

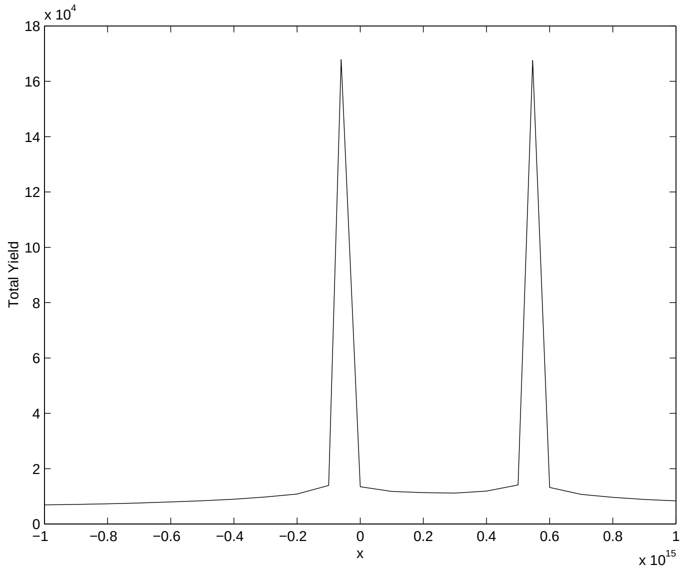

Figure 4.3: Total yield (with roots) of Polynomial A with $|x| \leq 10^{15}$

(a, b) at which *both* $F_1(a, b)$ and $F_2(a, b)$ are smooth (or almost smooth). In this case, since we use line sieving, $y = 1$. So far we have considered only the *yield* of f_1 and f_2 individually. It is reasonable to assume that these yields are independent. That being the case, the likelihood of a given $x = a$ causing both $f_1(a)$ and $f_2(a)$ to be simultaneously smooth will increase if the regions of maximal yield of f_1 and f_2 coincide. The same argument holds for sieving $F_1(x, y)$ and $F_2(x, y)$. Particularly if one is using more than two non-linear polynomials (Remark 2.3.1 and [32]) current performance might be exceeded by considering the proximity of the real roots of F_1 and F_2 when selecting polynomials.

Recall that in dealing with large prime yields we are assuming that the appearance of each prime in the factorisation of a given integer is independent. This of course is not true, and next we observe the effect of the dependence. Since this is of little practical consequence we leave it as an aside.

Aside 4.2.2 As before, let T be the total yield, and Q, R, S be the full, 1LP and 2LP yields respectively. For all five polynomials the proportions Q/T and R/T increase close to real roots at the expense of S/T. For example, for Polynomial A the proportion Q/T increases from 10% to 18%, R/T increases from 38% to 44% and S/T decreases from 52% to 38%. For the other polynomials the proportions take similar values.

Recall from Section 2.2.1 the generalisations $\rho_2(\mathbf{u})$ and $\rho_3(\mathbf{u})$ of $\rho(u)$. These functions describe the joint distributions for the two and three (respectively) largest prime factors of r as $r \to \infty$. We are interested in the special cases in which r has exactly one or exactly two prime factors at most B_2, but is otherwise B_1-smooth. Let $\rho_2(u, v)$

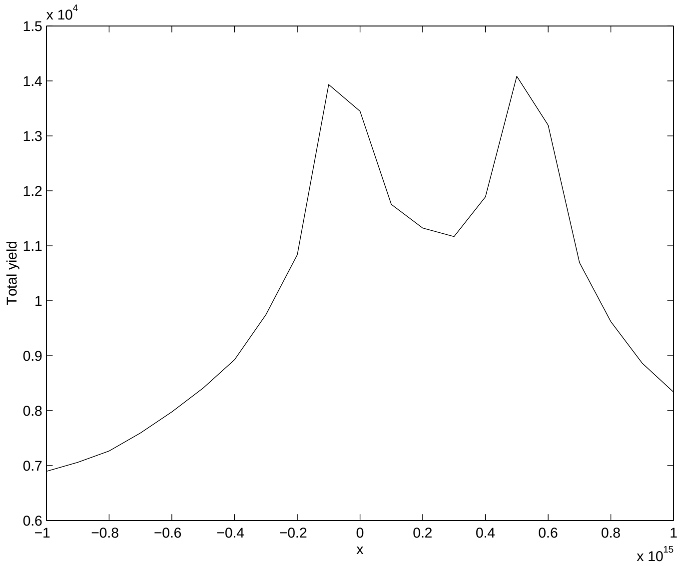

Figure 4.4: Total yield (without roots) of Polynomial A with $|x| \leq 10^{15}$

be the former function and $\rho_3(u, v)$ be the latter function, with $u = \log r / \log B_1$ and $v = \log r / \log B_2$.

Using the methods of [3] and [45] to calculate these functions, we observe that in the range of interest

$$\frac{\partial \rho_2}{\partial v} > \frac{\partial \rho_2}{\partial u} \quad \text{and} \quad \frac{\partial \rho_3}{\partial v} \gg \frac{\partial \rho_3}{\partial u}. \tag{4.13}$$

Note that the inequality for $\rho_2(u, v)$ is not true for arbitrary u and v. Intuitively (4.13) means that as r increases, the smoothness probabilities for 2LP-smoothness (and to a lesser extent 1LP-smoothness), depend more on r being B_2-smooth than on the cofactor (with the large primes removed) being B_1-smooth. That is, B_2-smoothness is the "difficult" property. The difference in (4.13) between ρ_2 and ρ_3 comes from

$$\frac{\partial \rho_2}{\partial u} > \frac{\partial \rho_3}{\partial u}.$$

Intuitively, ρ_2 ought to be more sensitive than ρ_3 to changes in u because a B_2-smooth integer with only one known prime factor between B_1 and B_2 is less likely to be otherwise B_1-smooth than one of the same size with two known prime factors between B_1 and B_2.

Now, since $B_1 < B_2$

$$\frac{du}{dr} > \frac{dv}{dr}. \tag{4.14}$$

Ignoring for the moment the question of root properties, (4.13) and (4.14) imply that as $|f(x)|$ decreases S/T ought to decrease relative to both Q/T and R/T, and that R/T ought to decrease slightly relative to Q/T.

4.2.2 Estimating Yield

Recall that with $f(x) > 0$ and

$$u_f(x) = \frac{\log f(x) + \alpha(f)}{\log B}$$

we assume that

$$P(f(x), B) \approx \rho(u_f(x)) + (1-\gamma)\frac{\rho(u_f(x)-1)}{\log f(x)}.$$

Suppose $I \subset \mathbb{Z}$ is some sieve interval. Then

$$\sum_{x\in I} P(f(x), B) \approx \sum_{x\in I}\left[\rho(u_f(x)) + (1-\gamma)\frac{\rho(u_f(x)-1)}{\log f(x)}\right]. \tag{4.15}$$

We use the right hand side of (4.15) to approximate the full yield of f across I.

In practice $|I|$ is large, so (4.15) is too time consuming to compute completely. Instead we approximate the summation by breaking I into K sub-intervals over which the right hand side of (4.15) does not change significantly. Let I_K be the interval I so divided, so I_K contains every $|I|/K$-th element of I. Hence, if X_f again denotes the full yield of f across I, then

$$X_f \approx \frac{|I|}{K}\cdot\sum_{x\in I_K}\left[\rho(u_f(x)) + (1-\gamma)\frac{\rho(u_f(x)-1)}{\log f(x)}\right]. \tag{4.16}$$

4.2.3 Examples

We now examine estimate (4.16) in the context of both peak yield and yield across the region. In the calculations below we use $K = 10^5$.

Peak Yields

We tested estimate (4.16) for X_f on seven polynomials with α-values sufficiently low to be acceptable number field sieve polynomials. In particular, we used Polynomial A, and six other polynomials $F, G, \ldots, K$. Polynomials $F, \ldots, K$ are polynomials used to factorise 105, 106 and 107 digit integers in [33]. Details of each are in Appendix A.

We calculated estimate (4.16) in an interval of size 10^8 across one real root of each polynomial, and sieved the polynomial across the same root. Yields across the two roots of each polynomial are almost identical so the choice of root is arbitrary. We used $B = 1600000$ for polynomials F and G in accordance with [33], otherwise $B = 2700000$. Table 4.8 contains the results for full relations.

The estimate places only one polynomial, J, in the incorrect position, and has an average relative error of 5.9% (most of which is contributed by polynomials J and F).

Poly-nomial	Est. full yield	Full yield	Relative error (%)
K	30462	30732	-0.9
J	30461	26100	16.7
A	30193	29005	4.1
H	27621	28248	-2.1
I	25583	24646	3.2
F	25209	22186	13.6
G	17096	15989	6.9

Table 4.8: Estimated vs actual full yield

Yield Across the Sieving Region

Table 4.8 tests only the peak yields of the polynomials. We saw at Remark 4.2.1 that it is also of interest to note how estimate (4.16) changes across an entire sieve interval. In Figure 4.5 below we show estimate (4.16) across the entire $|x| \leq 10^{15}$ interval, at uniformly spaced sub-intervals, for Polynomial A. We also show estimate (4.16) at $\alpha = 0$, that is, the expected yield if values taken by Polynomial A are as likely to be smooth as random integers of the same size. This is much lower than the actual yield.

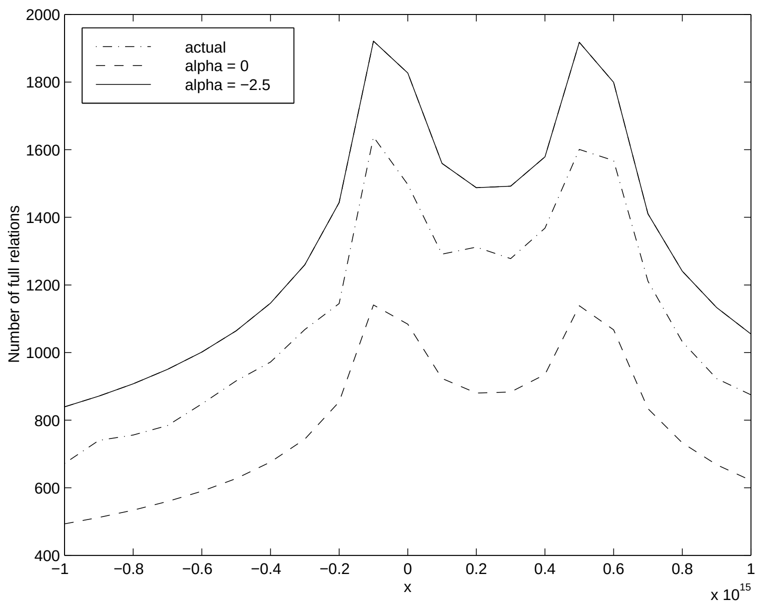

Figure 4.5: Estimated and actual yield of Polynomial A with $|x| \leq 10^{15}$

We conclude that the approach described in Section 4.2.2 to estimating yield is useful.

4.2.4 Polynomials for Larger N

We now extend the approach in (4.16) to consider yield due to root properties in the context of larger N. It is not clear in advance that we should expect differences in yield similar to those exhibited on quadratic polynomials. Not only are the integers which are required to be smooth larger, but B also is larger, and the practical range of α values is different.

The non-linear polynomial considered is now the homogeneous polynomial $F(x, y)$, so its sieve region lies properly in the x, y-plane. In fact as we see in the next chapter, the region is usually a rectangle much longer (x direction) than it is wide (y direction). We denote the length to width ratio s, and in this section will consider a fixed sub-rectangle S of the entire sieve region which also has length to width ratio s.

Again we let X_F denote the full yield of F over S. This yield depends on $\alpha(F)$, and in this section the aim is to compute

$$Q(\alpha) = \frac{X_F(\alpha)}{X_F(0)}.$$

To compute $Q(\alpha)$ we divide S into K equally sized sub-rectangles labelled S_i for $i = 1, \dots, K$. Let F_i be the mean value of $F(x, y)$ across S_i. Putting

$$u_i(\alpha) = \frac{\log F_i + \alpha(F)}{\log B}$$

we obtain for the probability, depending on α, that F_i is B-smooth

$$P_\alpha(F_i, B) \approx \rho(u_i(\alpha)) + (1-\gamma)\frac{\rho(u_i(\alpha) - 1)}{\log F_i}.$$

Assume the distribution of coprime integer pairs (x, y) is uniform throughout S. That, and the fact that all S_i have the same area imply that

$$Q(\alpha) \approx \frac{\sum_{i=1}^{K} P_\alpha(F_i, B)}{\sum_{i=1}^{K} P_0(F_i, B)}. \tag{4.17}$$

Here we calculate (4.17) using parameters from the factorization of RSA-140. The non-linear polynomial used in the factorisation is

$$\begin{aligned} F_1(x, y) \;=\; & \;\;439682082840\,x^5 \\ & +390315678538960\,x^4 y \\ & -7387325293892994572\,x^3 y^2 \\ & -19027153243742988714824\,x^2 y^3 \\ & -63441025694464617913930613\,x y^4 \\ & +318553917071474350392223507494\,y^5. \end{aligned}$$

We examine this polynomial in more detail in the next two chapters. For now all that is relevant is that the sieve region that best fits F_1 has $s \approx 4000$. For the calculations shown below we used a rectangle S of area 10^8 with $s = 4000$, centred on the y-axis at $y = 5000$. This places S in a typical portion of the entire sieve region. We used $K = 10^5$ and $B = 2^{24} - 1$ (see Section 6.2).

The practical range for α we take to be $[-7, 0]$. This is much more extreme than in the case of the quadratic polynomials of the previous sections. With the degree d as high as $d = 5$ we find polynomials with much better root properties than when $d = 2$. This is due partly to higher degree polynomials having the capacity to have more roots for each prime $p > d$, but mainly to extra tricks we have which rely on d being at least four. We explain these tricks in the next chapter.

For now we use F_1 only as a source F-values typical of those required to be smooth for factorisations of large N. Figure 4.6 below shows $Q(\alpha)$ computed using (4.17) on F_1 with α in the practical range.

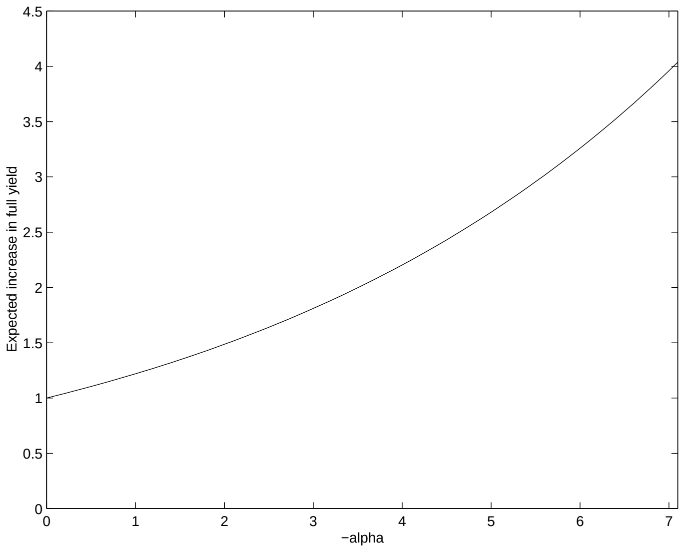

Figure 4.6: Full yield due to root properties at $N = $ RSA-140

4.3 Summary

In this chapter we have investigated the yield of number field sieve polynomials. We take yield to be influenced by two factors, size and root properties. The effect of root properties on yield has not previously been well understood. Here we demonstrate that, for example, under the conditions of large RSA moduli factorisations root properties can influence yield by up to a factor of 4.

Since it is therefore clear that root properties ought to be taken into account when modelling yield, we give a simple estimator of yield which combines both root properties and size. We then compare estimated yields to actual yields. Our estimate quantifies yield within an accuracy sufficient for our purposes, both absolutely and relatively.

We are hence led to an understanding of yield: what should be sought is a good *combination* of size and root properties. We turn now to the problem of finding polynomials with such combinations.

Chapter 5

Finding Good Polynomials

In this chapter and the next we employ our understanding of polynomial yield in selecting polynomials for factorisations of large general integers. This chapter contains descriptions of the relevant computations, and the next chapter contains the examples RSA-130, RSA-140 and RSA-155.

We consider the selection problem in two stages. In the first stage, we generate a large sample of good polynomials. This process is described in Section 5.1. Thousands of polynomials survive this stage, so sieving experiments are still impracticable. However there remains significant variation in yield across this sample. Thus in the second stage we identify, *without* sieving, the best polynomials in the sample. This process is described in Section 5.2. Section 5.3 contains a summary.

Throughout this chapter we distinguish two types of polynomial, namely *non-skewed* and *skewed*. The traditional approach to polynomial selection for RSA factorisations is to search for polynomials all of whose coefficients are small, without regard to root properties. All coefficients being small endears polynomials to sieving regions $-U \leq x \leq U$ and $1 \leq y \leq U$, for some integer U. We refer to these polynomials as non-skewed. In this chapter we extend this approach by giving simple methods of finding non-skewed polynomials with good combinations of size *and* root properties. Section 5.1.1 describes generation of many good non-skewed polynomials, and Section 5.2.1 describes identification of the best ones. We demonstrate the strength of even these simple methods in the next chapter by repeating the polynomial selection for the RSA-130 factorisation.

The non-skewed case is also a useful introduction to more complicated methods of finding good skewed polynomials. In the case of skewed polynomials, we require only some of the coefficients to be small. The coefficients a_d, a_{d-1} and a_{d-2} will be particularly small, and usually the coefficients will increase in absolute value from a_d through a_0. The natural sieving region for skewed polynomials is a rectangle S whose length (x-direction) to width ratio is s, with $s > 1$. We fit a different S to each polynomial. In practice we encounter s values up to approximately 10^6. Indeed, we go to some effort to construct *highly* skewed polynomials. There are implementation specific reasons for seeking such polynomials. More importantly though we are able to introduce additional techniques to find highly skewed polynomials with excellent root properties.

We describe the generation of such polynomials in Section 5.1.2. Isolating the best skewed polynomials requires only simple adjustments to the procedure for non-skewed polynomials, and we discuss this in Section 5.2.2. We demonstrate methods for finding skewed polynomials in the next Chapter by describing the polynomial selection for the factorisations of RSA-140 and RSA-155.

5.1 Generating Good Polynomials

Recall from Chapter 2 that for integers of the size under consideration, the base-m method is the best method we have of choosing polynomials. So d is fixed and we seek $m \approx N^{1/(d+1)}$ with a polynomial f of degree d for which

$$f(m) \equiv 0 \bmod N. \tag{5.1}$$

Sieving occurs over the polynomials $F_1(x, y) = y^d f(x/y)$ and $F_2(x, y) = x - my$.

As we have seen before, the polynomial f descends from the base-m representation of N. Let the coefficients of this expansion be $a_i^{(m)}$. That is,

$$N = \sum_{i=0}^{d} a_i^{(m)} m^i$$

with $0 \leq a_i^{(m)} < m$. Heuristically it is sensible to adjust the $a_i^{(m)}$ to lie between $-m/2$ and $m/2$. In fact, if $a_i^{(m)} > \lfloor m/2 \rfloor$ then we replace $a_i^{(m)}$ with $a_i^{(m)} - m$ and $a_{i+1}^{(m)}$ with $a_{i+1}^{(m)} + 1$. Let

$$f_m(x) = \sum_{i=0}^{N} a_i x^i$$

be the polynomial whose coefficients are the $a_i^{(m)}$ reduced in this way, working from $i = 0, \ldots, d$ through the coefficients.

The exercise now is to choose m and f_m (or some variant thereof which preserves (5.1)) with good combinations of size and root properties. In the case of non-skewed polynomials, we consider only f_m. In the case of skewed polynomials, we have the freedom to explore many variants of f_m.

5.1.1 Non-skewed Polynomials

Here we give simple methods for choosing good non-skewed f_m. Even these simple methods suffice to give significant improvements over previous factorisation efforts.

We consider first the problem of generating f_m which are "small", then of generating f_m with better than average root properties.

What does it mean for f_m to be small?

Definition 5.1.1 *A base-m representation f_m is χ-small when χ is the largest value of $|a_i|/m$ for $i = 1, \dots, d-1$.*

We refer to these simply as *small base-m representations* if the value of χ is not material.

A necessary condition on a particular representation being small is that the coefficients a_d and a_{d-1} are small. By the choice of m it is easy to ensure that a_d is small. Our search simply employs the fact that for small a_d, small $a_{d-1}^{(m)}$ occur only when m is close to a value at which a_d changes.

Example 5.1.2 Let $N = 9999399973 = \text{NextPrime}(10^5) \times \text{PreviousPrime}(10^5)$. The following is the sequence of (unreduced) base-m representations of N around the value of m which forces a_3 to decrease from 9 to 8. The coefficients are listed as $[a_0, \dots, a_3]$. Notice the (almost) linear change in a_2.

m	a_i
1030	[323, 405, 155, 9]
1031	[64, 122, 128, 9]
1032	[61, 925, 100, 9]
1033	[260, 751, 73, 9]
1034	[607, 631, 46, 9]
1035	[13, 566, 19, 9]
1036	[493, 554, 1028, 8]
1037	[959, 596, 1002, 8]
1038	[319, 693, 976, 8]
1039	[594, 843, 950, 8]
1040	[693, 7, 925, 8]
1041	[562, 264, 899, 8]
1042	[147, 575, 873, 8]

Table 5.1: A sequence of base-m representations

For fixed m, we now consider the coefficients $a_i^{(m+k)}$ of the base-$(m+k)$ expansion of N, as functions of $a_i^{(m)}$ and k. The coefficients are related by the fact that

$$\sum_{i=0}^{d} a_i^{(m)} m^i \;=\; \sum_{i=0}^{d} a_i^{(m+k)} (m+k)^i \;=\; N.$$

Matching the coefficients of the polynomials $\sum_{i=0}^{d} a_i^{(m)} x^i$ and $\sum_{i=0}^{d} a_i^{(m+k)} x^i$ reveals that

$$a_i^{(m+k)} \equiv \sum_{j=i}^{d} a_j^{(m)} \binom{j}{j-i} (-k)^{j-i} \mod (m+k) \tag{5.2}$$

for $i = 1, \dots, d-1$. For a_{d-1} this means that

$$a_{d-1}^{(m+k)} \equiv \left(a_{d-1}^{(m)} - dk a_d^{(m)} \right) \mod (m+k). \tag{5.3}$$

The value of m which causes the leading coefficient to decrease from a_d to $a_d - 1$ is given by

$$m_1 = \left\lfloor \left(\frac{N}{a_d} \right)^{\frac{1}{d}} \right\rfloor .$$

From (5.3) the values of m surrounding m_1 for which

$$|a_{d-1}^{(m)}| \leq \chi m \tag{5.4}$$

are easily determined. Moreover, the proportion of m-values satisfying (5.4) is approximately 2χ . Hence, compared to choosing m at random, conditioning the search on m guaranteed to give $|a_{d-1}| \leq \chi m$ increases its efficiency by a factor of approximately $1/2\chi$. Typically we use $\chi = 0.02$, so the search efficiency increases by a factor of 25.

This method does not give much information on the location of m for which small values of lower order coefficients must lie. For example, the third coefficient of a quintic representation is

$$a_3^{(m+k)} \equiv \left(10 a_5^{(m)} k^2 - 4 a_4^{(m)} k + a_3^{(m)} \right) \mod (m+k),$$

which in practice means that the change in $a_3^{(m+k)}$ as a function of k is no longer sufficiently small to be useful.

Consider now the problem of generating non-skewed polynomials with better than average root properties. Recall that we regard $F_1(x, y)$ as having two types of roots modulo p, projective and non-projective. Here we equip F_1 with better than average root properties by forcing it to have good projective roots modulo small p^k. The appearance of good non-projective roots at this stage we leave to chance.

The following example illustrates the effect of this observation on the distribution of root properties amongst polynomials examined.

Example 5.1.3 Let $N =$ RSA-140. A reasonable range of leading coefficients of non-skewed f_m for N is $[10^{20.3}, 10^{21.3}]$. We choose a_d in this range to contain a cofactor c in each of the following five cases:

1. the worst case, a_d is prime,

2. the average case, a_d is chosen uniformly at random,
3. a good case, $c|a_d$ with $c = 2 \cdot 3 \cdot 5 \cdot 7$,
4. a better case, $c = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29 = 10^{9.8}$, and
5. an even better case, $c = 2^5 \cdot 3^4 \cdot 5^3 \cdot 7^3 \cdot 11^2 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29 = 10^{16.6}$.

Randomly chosen samples of 100 polynomials in each of the five cases above gave the following α values.

Case	Min $\alpha(F)$	Mean $\alpha(F)$	Max $\alpha(F)$
1	-0.67	1.16	2.63
2	-1.62	0.44	2.33
3	-3.02	-0.77	0.77
4	-3.25	-1.40	0.15
5	-3.54	-1.88	-0.47

Table 5.2: Polynomials with many small projective roots

Despite c being large in case 5, it is still sufficiently small to allow examination of many polynomials.

Note that computing the ideal decomposition for ideals corresponding to projective roots requires more effort than those corresponding to non-projective roots. Hence polynomials found by this method will require marginally more effort in the square root stage, but the benefit far outweighs this extra cost.

Our procedure for finding non-skewed polynomials is the following.

Procedure 5.1.4 (Non-skewed Base-m Polynomials)

1. Fix an interval a_d in which each a_d is significantly smaller than its corresponding m. In fact, select χ_1, χ_2 for which a_d will satisfy $\chi_1 \leq |a_d|/m \leq \chi_2$. The interval for a_d is then bounded below and above by
$$\log a_d = \frac{(d \log \chi_j + \log N)}{d+1}$$
at $j = 1, 2$ respectively. This corresponds to a range of m values bounded below and above by
$$\log m = \frac{(\log N - \log \chi_j)}{d+1}$$
at $j = 2, 1$ respectively.

2. Fix a cofactor c of a_d, with c a product of many small p^k. Of course several c will be used. For each a_d divisible by c in the interval described in Step 1, determine from (5.3) the values of m for which $|a_{d-1}/m| \leq \chi m$ (with $\chi \geq \chi_2$).

3. For each m identified in Step 2, check the remaining coefficients of f_m. If f_m is χ-small, compute an approximation to $\alpha(F_1)$, and if that is also sufficiently small, output f_m.

5.1.2 Skewed Polynomials

We turn to our more involved procedure for finding good skewed polynomials. It is not just skewed, but *highly* skewed polynomials that are the real target of this section, because we have a procedure for finding descendants of highly skewed F_1 with excellent root properties. The following example illustrates a highly skewed target.

Example 5.1.5 Let N = RSA-140. We give two pairs of polynomials for N. The pair (F_1, F_2) is the pair of skewed polynomials used for the factorisation of RSA-140. The pair (G_1, G_2) is the best non-skewed pair identified during the search for RSA-140 polynomials.

$$\begin{aligned}
F_1(x,y) \quad = \quad & 439682082840\,x^5 \\
& +390315678538960\,x^4 y \\
& -7387325293892994572\,x^3 y^2 \\
& -19027153243742988714824\,x^2 y^3 \\
& -63441025694464617913930613\,xy^4 \\
& +318553917071474350392223507494\,y^5 \\
\\
F_2(x,y) \quad = \quad & x - 34435657809242536951779007\,y \\
\\
G_1(x,y) \quad = \quad & 2378666111034213000000\,x^5 \\
& -5148567155828225103040\,x^4 y \\
& -47226689253467208438840\,x^3 y^2 \\
& +6545365626333869758617\,x^2 y^3 \\
& -3356924353646091366162\,xy^4 \\
& -5142225622472630020004\,y^5 \\
\\
G_2(x,y) \quad = \quad & x - 61711974230444693875191\,3\,y.
\end{aligned}$$

The recommended sieve rectangle for F_1 has $s = 4096$. Both F_1 and G_1 are unusually small over their respective regions ($\chi(G_1) = 0.011$). Both also have good root properties, but the main difference between the polynomials is in just how good their root properties are. We have $\alpha(F_1) = -7.0$ and $\alpha(G_1) = -4.2$.

How does this difference in root properties come about? Notice that $|a_1|$ and $|a_0|$ of F_1 are larger than m. Clearly on construction of f_m they cannot start that way. In fact we adjust f_m to cause it to appear highly skewed, compensating for large low order (in x) coefficients by skewing the sieve region. Once the low order coefficients are large we try many further adustments hoping for some which lead to polynomials with excellent root properties.

During the skewing process f may move "off-centre" along the x-axis. We also make an adjustment that fixes this. Hence, we adjust f in two ways:

- Translation by t : $f_t(x) = f(x-t)$ for some $t \in \mathbb{Z}$. This leaves root properties unaltered but can improve size. With $m_t = m + t$ we preserve (5.1).
- "Rotation" by P: $f_P(x) = f(x) + P(x) \cdot (x - m)$ for some polynomial P whose degree is small compared to d. This preserves (5.1). Rotation by P can alter both size and root properties. Presently we use only linear P, but for higher degree f, polynomials, higher degree P could be used without impinging on the high order coefficients of f.

Translation by t need not be peculiar to skewed polynomials, but we make more use of it here than with non-skewed polynomials. Most of the benefit comes from rotations by well chosen P. Indeed we use two rotation steps. The first is aimed at producing highly skewed f which are unusually small over some skewed rectangle. The second is aimed at taking these f and rotating them to form new ones which, whilst retaining desirable size properties of the old ones, also have excellent root properties. Our implementation of this procedure is due mainly to Peter Montgomery.

We describe this procedure in four steps. In Steps 1 and 2 we isolate skewed polynomials which are unusually small over some rectangle. Polynomials surviving Step 2 enter Step 3. In Step 3 we seek rotations giving polynomials with excellent root properties without destroying the good size properties inherited from Step 2. Step 4 produces the output.

Procedure 5.1.6 (Skewed Base-m Polynomials)

1. Find leading coefficients a_d divisible by many small p^k for which there exists a base m expansion with skewed coefficients. For each such a_d we examine
$$m \approx \left\lfloor \left(\frac{N}{a_d}\right)^{\frac{1}{d}} \right\rfloor.$$
Check the magnitude of a_{d-1}, and of a_{d-2} compared to m, by computing the integral and non-integral parts of
$$\frac{N - a_d m}{m^{d-1}} = a_{d-1} + \frac{a_{d-2}}{m} + O(m^{-2}).$$
If these are sufficiently small, accept a_d and m.

2. Compute some initial adjustments to f_m aimed at skewing it further and reducing its size over a new skewed rectangle. In particular, we consider variables c_1, c_0, t and s, the adjustments

- translation by t,
- rotation by $P(x) = c_1 x - c_0$,

and the rectangle with $|x| < \sqrt{s}$ and $|y| < 1/\sqrt{s}$. Call this rectangle S, and denote by the subscript t variables which have been translated by t. Now let

$$\begin{aligned} f(x) &= f_m(x_t) + (c_1 x_t + c_0)(x_t - m_t), \text{ and} \\ F(x,y) &= y^d f(x/y). \end{aligned}$$

At this stage we treat c_1, c_0 and t as real variables. We now apply a multi-variable minimization procedure to minimize

$$\int\!\!\int_S F^2(x,y)\,dxdy$$

with respect to c_1, c_0, t and s. The optimal values of c_1, c_0 and t are rounded to integers, and s is recomputed. The average log size $I(F,S)$ over the new S is estimated, with

$$I(F,S) = \log\left(\sqrt{\int\!\!\int_S F^2(x,y)\,dxdy}\right),$$

and if that is sufficiently small, we proceed to Step 3.

3. Search for polynomials with excellent root properties amongst polynomials with similar size properties to f. Let j_1, j_0 be integers with $|j_1| < J_1$ and $|j_0| < J_0$. Typically we have $J_1 \ll J_0$. We investigate the polynomials

   $$f_{j_1,j_0}(x) = f(x) + (j_1 x - j_0)(x - m)$$

   using a sieve-like procedure to identify j_1, j_0 pairs which ensure f_{j_1,j_0} has good root properties.

   We describe the sieve-like procedure. For each small prime p we consider contributions of p^k for $k \geq 1$. Take p^k and j_1 to be fixed, j_0 and l to be variable. The values $f_{j_1,j_0}(l) \bmod p^k$ can be computed quickly for successive $l = 0 \ldots p^k - 1$ by finite differences. For each such l we find, simply by solving a linear congruence, $j_0 \in \mathbb{Z}_{p^k}$ for which

   $$f_{j_1,j_0}(l) \equiv 0 \bmod p^k. \tag{5.5}$$

   For each solution j_0 of (5.5) we estimate $\mathrm{cont}_{p^k}(F_{j_1,j_0})$, and in an array of length p^k record $\mathrm{cont}_{p^k}(F_{j_1,j_0})$ in the position corresponding to j_0. We also record $\mathrm{cont}_{p^k}(F_{j_1,j_0})$ at any projective roots. On completion mod p^k, this array is replicated throughout the entire J_0-space.

   Once this is completed for all small p and all j_1, the values in the j_1, j_0 array approximate $\alpha(F_{j_1,j_0})$ over the small primes considered.

4. Since $I(F_{j_1,j_0}, S) \approx I(F, S)$ we already have an approximation to the average size of F_{j_1,j_0}. So we take the initial rating of each F_{j_1,j_0} to be

$$I(F_{j_1,j_0}, S) + \alpha(F_{j_1,j_0}).$$

For those F_{j_1,j_0} whose initial rating is sufficiently low, we compute the coefficients of

$$f(x) + (j_1 x - j_0)(x - m),$$

and if it helps we compute translation of m and a new optimal value s for the translated F.

5.2 Isolating the Best Polynomials

From the procedures of the previous section we inherit a collection of many good polynomials. This collection may contain thousands of polynomials. Despite the fact that these are all good polynomials, there is still significant variation between best and worst yields in the collection. Indeed, the variation may exceed a factor of 50%. Since there are still too many polynomials to conduct sieving experiments, we require a fast and reliable procedure for rating the polynomials according to their yield, and therefore identifying the best ones.

Below we outline this procedure. For simplicity of explanation we consider the non-skewed case first. The skewed case is then a simple generalisation.

5.2.1 Non-skewed Polynomials

Consider first only the non-linear polynomial $F_1(x, y)$. Often with non-skewed polynomials, all considered values of m are similar, so the rating is determined mainly by F_1. Later we make trivial adjustments to consider F_2 also.

To ensure reliability of the rating it is crucial to have an accurate estimate of $\alpha(F_1)$. Hence, at this point we compute $\mathrm{cont}_p(F_1)$ for small p directly. That is, from a sample of F_1-values we count appearances of p^k for $k \geq 1$ and take $\mathrm{cont}_p(F_1)$ to be the mean number of appearances per value. This slows the procedure somewhat, but since small p^k make a large difference to yield and since usually some small p are not well behaved, the extra computation is important. In fact we compute $\mathrm{cont}_p(F_1)$ directly for $p < 100$, and estimate $\mathrm{cont}_p(F_1)$ for $100 < p < 2000$.

Now, since F_1 is homogeneous, in polar coordinates

$$F_1(x, y) = r^d F_1(\cos\theta, \sin\theta).$$

At fixed $\theta = \theta_i$ any two polynomials of degree d grow as the d-th power of r along θ_i. So the values $F_1(\cos\theta_i, \sin\theta_i)$ are the most relevant for rating the yields of these polynomials.

Hence we fix $r = 1$ and put

$$u_{F_1}(\theta_i) = \frac{\log|F_1(\cos\theta_i, \sin\theta_i)| + \alpha(F_1)}{\log B}.$$

We divide the interval $[0, \pi]$ uniformly into K sub-intervals and put

$$\theta_i = \frac{\pi}{K}\left(i - \frac{1}{2}\right)$$

for $i = 1, \ldots, K$. That is, θ_i is the mean value of θ on the i-th sub-interval. Now we compute

$$\mathbb{E}(F_1) = \sum_{i=1}^{K} \rho(u_{F_1}(\theta_i)) \tag{5.6}$$

and take $\mathbb{E}(F_1)$ to be an estimated rating of F_1. That is, polynomials are ranked in descending order of $\mathbb{E}(F_1)$ values. The value of K is not crucial to the comparison between polynomials, but we use $K = 1000$.

Now consider also the linear polynomial $F_2(x, y)$. Smoothness bounds for F_1 and F_2 may be different, so we denote by B_{F_j} the smoothness bound for F_j. With

$$u_{F_j}(\theta_i) = \frac{\log|F_j(\cos\theta_i, \sin\theta_i)| + \alpha(F_j)}{\log B_{F_j}}$$

for $j = 1, 2$ we take $\mathbb{E}(F_1, F_2)$ defined by

$$\mathbb{E}(F_1, F_2) = \sum_{i=1}^{K} \rho(u_{F_1}(\theta_i))\rho(u_{F_2}(\theta_i)) \tag{5.7}$$

to be the estimated rating of a given pair of polynomials. That is, pairs of polynomials are ranked in descending order of $\mathbb{E}(F_1, F_2)$ values.

Note 5.2.1 The values $\mathbb{E}(F_1)$ for single polynomials should be compared only between polynomials F_1 of the same degree, and $\mathbb{E}(F_1, F_2)$ values for pairs of polynomials should be compared only between pairs of polynomials which are pairwise of the same degree.

Note 5.2.2 We observe the ranking induced by $\mathbb{E}$ to be independent of variations in B. Amongst all polynomials this is not necessarily true, but amongst a set of candidate polynomials in practice we expect this to be the case. Hence for example, using sub-optimal smoothness bounds should not change the fact that a polynomial is "good".

5.2.2 Skewed Polynomials

Here we generalise the previous computation to give a fair profile of F_1 and F_2 across a skewed region whose length to width ratio is s. Note that since values of m amongst polynomials may now differ substantially, both F_1 and F_2 should be considered at all times.

Consider F_1 and F_2 around an ellipse of fixed area, whose major and minor axes are in the ratio s. In particular let $s_1 = \sqrt{s}$ and $s_2 = 1/\sqrt{s}$ and consider the ellipse

$$\begin{aligned} x &= s_1 \cos\theta \\ y &= s_2 \sin\theta \end{aligned}$$

for $\theta \in [0, \pi]$. We again divide the θ interval uniformly into K equal sub-intervals and take θ_i for $i = 1, \dots, K$ to be the mean value of θ in each sub-interval.

With

$$u_{F_j}(\theta_i) = \frac{\log |F_j(s_1 \cos\theta_i, s_2 \sin\theta_i)| + \alpha(F_j)}{\log B_{F_j}}$$

for $j = 1, 2$ we take $\mathbb{E}(F_1, F_2)$ defined by

$$\mathbb{E}(F_1, F_2) = \sum_{i=1}^{K} \rho(u_{F_1}(\theta_i))\rho(u_{F_2}(\theta_i)) \tag{5.8}$$

to be the rating of a given pair of polynomials. That is, pairs of polynomials are ranked in descending order of $\mathbb{E}(F_1, F_2)$ values.

For any given N, at most approximately twenty polynomial pairs with highest $\mathbb{E}$ ratings will then be subjected to short sieving experiments.

5.3 Summary

In this chapter we have described methods for finding good base-m polynomials. We consider both skewed and non-skewed cases. In each case we consider two problems.

The first is the problem of generating large samples of polynomials which are small and have good root properties. In the case of non-skewed polynomials, we look only amongst polynomials whose first two coefficients are known to be small, and which have many projective roots modulo small p^k. We leave the appearance of many non-projective roots modulo small p^k to chance. In the case of skewed polynomials, we look only amongst polynomials with skewed coefficients and unusually small average size over some skewed rectangle, and with many projective roots mod small p^k. Moreover, we make use of the fact that the last few coefficients of highly skewed polynomials are large, with an efficient method of isolating polynomials which also have good non-projective roots modulo small p^k.

Having generated a large sample of good polynomials the second problem becomes isolating, without sieving, the best polynomials in the sample. In both skewed and non-skewed cases we do this by profiling the smoothness probability of each pair F_1, F_2 (adjusted for root properties), across the appropriate sieve region.

Chapter 6

Polynomials for RSA Factorisations

This chapter is a report on polynomial selection for several large RSA factorisations. We use the factorisations to exemplify and investigate the techniques described in previous chapters.

The two new factorisations discussed here are RSA-140 and RSA-155. Recall from Section 1.2.4 the details of the RSA Factorisation Challenge. The factorisation of RSA-140, completed in February 1999 set a new general factorisation record. At the time of writing this thesis, sieving for the factorisation of RSA-155 is complete. We expect the new record to be announced by September 1999. In this chapter we also re-consider the previous record set in 1996, RSA-130, as a means of testing some of the procedures of previous chapters.

Throughout this chapter we examine polynomials from two perspectives, namely *local* and *global*. The local perspective involves comparing individual polynomials and their properties. The global perspective involves placing polynomials in the context of the space of available polynomials - for example by asking questions like "how do the yields of the polynomials we now find compare to the yields of randomly chosen polynomials?". We take both local and global perspectives on polynomials examined for all three integers RSA-130, RSA-140 and RSA-155, however we emphasize the local perspective for the first two and the global perspective for the last one.

In Section 6.1 we examine polynomial selection for RSA-130. We seek polynomials which, under the conditions used for that factorisation and reported in [23], would improve the sieving time. Working under the conditions used in the factorisation means that we should consider only non-skewed polynomials. Hence we test Procedure 5.1.4 and the $\mathbb{E}$ rating procedure of Section 5.2. Even using only these techniques, the improvement obtained in a comparatively tiny period of time is surprising. Ultimately we find that in a fraction of the time used for the actual RSA-130 polynomial search we identify several polynomials whose full yields are 1.5–2 times that of the polynomial used in the factorisation. We will consider briefly a global comparison, by comparing the yield of the best polynomial so obtained to that of a polynomial of average yield for RSA-130.

In Section 6.2 we turn to RSA-140. When actually conducting this search, we first considered using non-skewed polynomials found by Procedure 5.1.4. Once it was clear we could obtain polynomials with better root properties *and* good size by Procedure 5.1.6, we decided to use the skewed polynomials. We use the RSA-140 factorisation to examine locally the properties of the best skewed polynomials, and to compare locally the best skewed and non-skewed polynomials. Since the procedure to find the skewed polynomials was under development during the search, we do not have a sufficiently large sample of skewed polynomials for detailed global considerations, so we consider only the comparison of best skewed polynomials to average skewed polynomials.

Given the results achieved with RSA-140, we searched only amongst highly skewed polynomials for RSA-155. The results are discussed in Section 6.3. We conducted a more comprehensive search, indeed several users ran the search program. It is timely to mention that we are particularly grateful to Arjen Lenstra for porting the search code to use his multiple precision arithmetic package LIP. That allowed other users to run it. Bruce Dodson ran several search jobs for RSA-155 polynomials, and the polynomial chosen for the factorisation appeared from one of his searches.

Since we therefore have a large and "stable" sample of good polynomials for RSA-155, we are able to make a more detailed global examination of the polynomials we find using Procedure 5.1.6. We do this in Section 6.3, as well as examining the top few polynomials locally. The global comparsions we make are aimed at

- placing the sample of polynomials generated during the search in the context of randomly generated polynomials, and
- examining the trade-off between polynomial search time and the corresponding saving in sieving time.

Section 6.4 contains a summary of this chapter.

All polynomials which are not given explicitly in the text of this Chapter are given in Appendix B.

6.1 RSA-130

By re-examining the polynomial selection task for the factorisation of RSA-130 we aim to test Procedure 5.1.4 for finding good non-skewed polynomials, and the ratings $\mathbb{E}(F_1, F_2)$ and $\mathbb{E}(F_1)$ of Section 5.2. We discuss three sets of polynomials, P_i, Q_i, and R_i. The P_i polynomials are the actual candidates discussed in [23], the Q_i are a better set of candidates we generated, and the R_i are the best candidates we generated.

6.1.1 The Fifteen Candidate Polynomials

The paper [23] describes a set of fifteen candidate polynomials considered for the factorisation of RSA-130. These fifteen polynomials were generated over some time, and identified as being good candidates on the basis of a rating measuring the size of the values taken by the polynomials [37]. The polynomials are labelled 1–15 according to this initial ranking, Polynomial 1 being the best ranked polynomial and Polynomial 15 the worst. According to [23] all fifteen candidates were then subjected to extensive sieving experiments and ranked according to their "true" yield as revealed by those experiments. The sieving experiments measured the yield of each *pair* of polynomials F_1, F_2. The two left-most columns (reproduced from [23]) of Table 6.1 show the rank and relative yields according to these experiments (we will refer later to the two right-most columns of this table).

[23] yield (%)		F_1 yield (%)	
14	100.0	14	100.0
4	99.1	4	98.6
1	93.7	1	90.2
12	87.5	12	87.3
8	82.2	8	77.7
3	80.0	2	74.7
10	77.8	9	73.4
2	76.8	11	73.0
11	76.6	10	72.7
15	75.9	3	72.3
9	75.4	15	71.6
5	70.1	5	67.9
7	64.9	13	61.4
13	64.2	7	60.0
6	57.8	6	52.9

Table 6.1: Sieving the RSA-130 polynomials

We refer to Polynomials $1, \dots, 15$ as $P_1, \dots, P_{15}$. Polynomial P_{14} was selected for use in the factorisation of RSA-130, and it becomes the polynomial we use as a benchmark for the polynomials we find.

Since they were selected on the basis of their size, the fifteen candidate polynomials all have unusually small coefficients. The largest value of χ for any of these polynomials is $\chi = 0.004$. However, they have a generally poor set of root properties. Table 6.2 gives the values $\alpha(F_1)$ for $P_1, \dots, P_{15}$.

Most of the candidate polynomials have $\alpha(F_1) > 0$, so for these polynomials sieving

$F_1(x,y)$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\alpha(F_1)$	1.09	1.15	1.96	0.58	1.50	2.15	2.60	0.62	0.20	0.78	1.04	-0.01	1.88	-0.40	0.44

Table 6.2: Root properties of the RSA-130 polynomials

is being conducted over integers less likely to be smooth than random integers of the same size. The reason P_{14} performs much better in sieving experiments than expected on the basis of its size alone, is that $\alpha(P_{14})$ is signficantly smaller than for the other polynomials. Actually $\alpha(P_{14}) < 0$ not because P_{14} has many non-projective roots modulo small primes, but because the leading coefficient of P_{14} has many small prime factors. Indeed

$$a_5 = 5748302248738405200 = 2^4 \cdot 3^4 \cdot 5^2 \cdot 19^2 \cdot 331 \cdot 114213131.$$

That is, P_{14} has many projective roots for small p.

6.1.2 On the Reliability of $\mathbb{E}$

We now use the fifteen polynomials from [23] and more of our own to test the reliability of $\mathbb{E}$ as a pre-sieving yield rating procedure.

Remark 6.1.1 Clearly the initial ranking of [23] based on size alone is inaccurate. Another method is mentioned without detail in [23] as being devised *after* the factorisation and giving reasonable correlation with the true rank for that set of polynomials. That method is due to Peter Montgomery and involves estimating the root mean square of each polynomial in some region and subtracting an estimate of the contribution of the small primes to that value. Indeed, we use an estimate similar to this as a first filter in Procedure 5.1.6, and we will see that again in Section 6.2. Our $\mathbb{E}$ method can be viewed as an extension of this method. The key differences are that we make a more accurate assessment of the effect of root properties, and we use the ρ function to emphasize regions of high smoothness probability where the polynomial takes smaller values.

Table 6.3 gives the rankings on the fifteen candidate polynomials of [23] under our $\mathbb{E}(F_1, F_2)$ ranking and under application of Montgomery's method of Remark 6.1.1.

Actual Yield		[23] rank		$\mathbb{E}$ rank	
Polys	Rank	Polys	Rank	Polys	Rank
14	1	4	2	14	1
4	2	14	1	12	4
1	3	8	5	4	2
12	4	10	7	1	3
8	5	15	10	9	11
3	6	1	3	3	6
10	7	12	4	2	8
2	8	3	6	8	5
11	9	9	11	15	10
15	10	2	8	10	7
9	11	11	9	11	9
5	12	7	13	5	12
7	13	13	14	13	14
13	14	5	12	7	13
6	15	6	15	6	15
		$r = 0.86$		$r = 0.91$	

Table 6.3: Rankings of the 15 candidate RSA-130 polynomials, $\mathbb{E}(F_1, F_2)$

The first column gives the true ranking revealed by the sieving experiments in [23]. Its left sub-column contains the polynomial labels, and its right sub-column contains the rank. So P_{14} is ranked first, and P_6 is ranked fifteenth. The second column gives the ranking induced by the method mentioned in [23]. Its left sub-column lists the polynomial labels in the order in which they are ranked, and its right sub-column gives the true rank of each polynomial. So this method ranks P_4 in first position (but P_4's true yield ranks second), P_{14} in second position (but P_{14}'s true yield ranks first), and so on. The third column gives the ranking induced by $\mathbb{E}$, listed in the same fashion as the second column.

One measure of the reliability of a ranking is its correlation coefficient with the true ranking, as defined in [23]. The correlation with the true rank for the method of [23] is $r = 0.86$, and for the $\mathbb{E}$ method $r = 0.91$. That is not a dramatic improvement, but the $\mathbb{E}$ ranking does seem to be more successful as a predictor, in the sense that it identifies better the very best polynomials. The trade-off of course is that our method is slightly more time consuming to compute. Indeed as indicated in Section 5.2 a method similar to that described in [23] is used *before* $\mathbb{E}$ to screen out the very worst polynomials.

6.1.3 Eighteen Different Candidates

We now introduce a new set of candidate polynomials labelled $Q_1, \ldots, Q_{18}$. These polynomials were generated using only the observation of Section 5.1.1 which isolates polynomials with small a_d and a_{d-1}, then screening by $\mathbb{E}$. We did not force these polynomials to have highly smooth a_d; root properties were left entirely to chance.

The purpose of exhibiting these polynomials is two-fold:

- to show that even leaving root properties to chance, significant improvements can be made provided we know what to look for (that is, having a reliable pre-sieving rating procedure)
- to give another set of test polynomials on which the $\mathbb{E}$ rating ought to work reliably.

The relative yields of the polynomials Q_i were determined by the sieving experiments described in the following remark.

Remark 6.1.2 We sieved across the entirety of the rectangle $-10^4 \leq x \leq 10^4$ and $1 \leq y \leq 10^4$, using only the quintic polynomial F_1. Omitting the linear polynomial makes the experiments much quicker, and does not significantly affect the outcome since all values of m used are similar. Furthermore, in the first instance we sieved only for full relations ($B = 11380951$). To verify the reliability of these smaller and restricted sieving experiments, we also sieved each of the quintic candidate polynomials $P_1, \ldots, P_{15}$ in this way. The results form the right hand columns of Table 6.1. The correlation coefficient of the ranking according to our experiments and the ranking according to experiments of [23] is 0.92. The differences appear only where the relative yields are very close. The full yield of P_{14} we obtained is 15990 relations - from this the relative yields below can be placed in perspective.

The left column of Table 6.4 lists the polynomials labelled $1, \ldots, 18$ in the order in which their yields appear from our sieving experiments. Its right sub-column gives the full yield relative to the benchmark P_{14}. So the best polynomial found by these primitive means has a full yield 47% better than that used to factorise RSA-130. The middle and right hand columns give the rankings of these polynomials by the method of [23] (adjusted to consider only the quintic polynomial) and the ranking $\mathbb{E}(F_1)$ (although as we would expect, adjusting the rankings to take into account only the non-linear polynomial makes little difference). The former method has correlation $r = 0.53$ and the latter $r = 0.91$ with the ranking revealed by the sieving experiments. Notice that $\mathbb{E}$ again isolates the very best polynomials reliably. Since the range of relative yields of $Q_1, \ldots, Q_{18}$ is smaller than that in $P_1, \ldots, P_{15}$, we consider them a more difficult set of polynomials to rank well.

Actual Yield		[23]	$\mathbb{E}$
Rank	cf Poly P_{14}	Rank	Rank
1	1.47	5	1
2	1.37	6	3
3	1.34	2	6
4	1.33	4	7
5	1.31	8	2
6	1.30	12	8
7	1.28	17	4
8	1.27	13	10
9	1.26	11	5
10	1.23	1	9
11	1.23	18	15
12	1.23	9	11
13	1.22	14	18
14	1.16	10	13
15	1.16	3	12
16	1.16	15	14
17	1.15	7	17
18	1.14	16	16
		$r = 0.53$	$r = 0.91$

Table 6.4: Rankings of 18 better RSA-130 polynomials, $\mathbb{E}(F_1)$

Amongst the polynomials generated during this search, we found several with coefficients as small as those in $P_1, \ldots, P_{15}$. However, none of these polynomials have yields significantly better than P_{14}. Instead, the better polynomials here all have coefficients significantly larger than those of $P_1, \ldots, P_{14}$ but better root properties. Polynomial Q_1 for example as $\chi = 0.016$. Table 6.5 shows the root properties of $Q_1, \ldots, Q_{18}$.

Of course the polynomials Q_i are of no practical use, because their yields are poor compared to the polynomials we exhibit next.

6.1.4 Good Polynomials for RSA-130

We conducted a brief search for RSA-130 polynomials using the entirety of Procedure 5.1.4, combined with the $\mathbb{E}$ ranking procedure.

Table 6.6 shows the full yields of the best polynomials identified in this search, relative to the yield of P_{14} and according to experiments conducted as in Remark 6.1.2. We exhibit also values of α and χ for each polynomial.

So the best polynomial identified by this method has a full yield twice that of the polynomial used to factorise RSA-130.

$F_1(x,y)$	1	2	3	4	5	6	7	8	9
$\alpha(F_1)$	−3.62	−2.45	−2.15	−2.70	−2.15	−2.01	−2.19	−2.68	−1.93

...	10	11	12	13	14	15	16	17	18
...	−2.03	−2.43	−2.24	−2.08	−1.82	−1.71	−0.95	−2.10	−1.87

Table 6.5: Root properties of $Q_1, \ldots, Q_{18}$

Poly.	Yield cf P_{14}	α	χ
1	2.00	−3.88	0.011
2	1.98	−3.92	0.015
3	1.89	−3.94	0.017
4	1.66	−4.00	0.018
5	1.63	−4.30	0.016
6	1.48	−3.20	0.014

Table 6.6: Good RSA-130 polynomials

We performed more experiments to confirm the yield of polynomial R_1. Sieving for full *and* large prime relations using $B_1 = 11380951$ and $B_2 = 120000000$ in accordance with [23] revealed a total yield (the sum of full yield, 1LP- yield and 2LP- yield) 1.83 times that of polynomial P_{14}. Hence the true benefit obtained from R_1 could be anything from a factor of approximately 1.8 to 2, depending on the particulars of the sieving technique. We also repeated this experiment using both the linear and algebraic polynomials in each case, using $B_1 = 3497867$ and $B_2 = 120000000$ for the linear polynomial in accordance with [23]. We again found that the total yield of R_1 is 1.83 times that of P_{14}.

Compare the α values of the polynomials R_i to those of the polynomials Q_i. The difference is of course due to the forced projective roots in the R_i polynomials. Another illustration of this effect is Example 5.1.3.

Figure 6.1 shows values of the homogeneous polynomial $R_1(x, y)$ over a portion of the sieve region. The portion is $-10^6 \le x \le 10^6$ and $1 \le y \le 10^6$. The diagonal lines eminating from the origin are the three real roots x/y of R_1. Most of the relations of course will come from the darker “valleys” carved by the real roots.

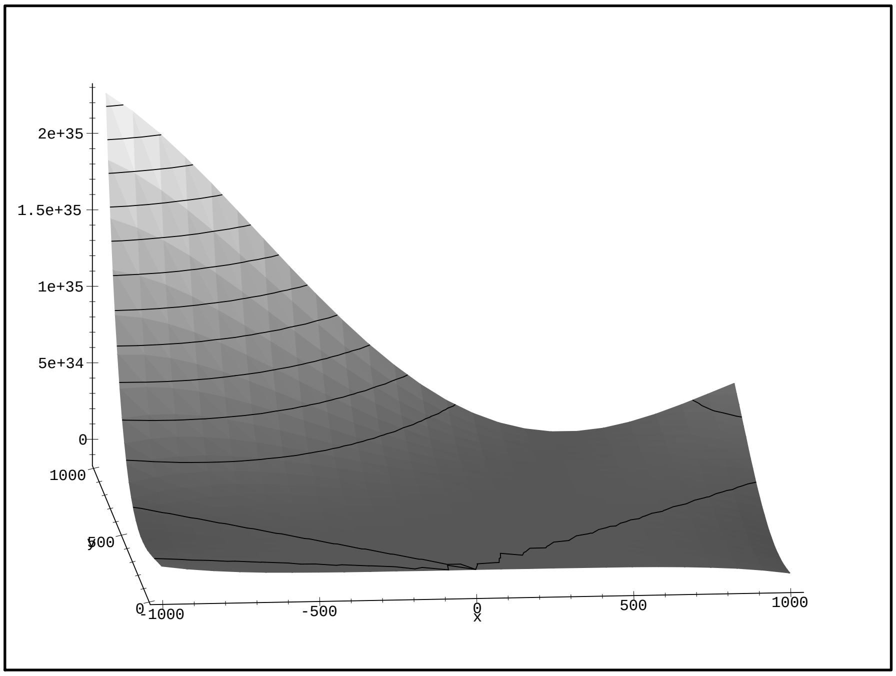

Figure 6.1: Polynomial R_1

Finally in this section we make a brief global observation by comparing the yield of R_1 to that of a polynomial of average yield. To find a polynomial of average yield we chose 100 base-m representations by choosing m uniformly at random in the relevant range. We then took a polynomial of average yield to be the polynomial in this sample whose $\mathbb{E}$ rating was closest to the mean rating of the sample. This suffices as an approximation to a polynomial of average yield, and saves sieving a large sample of polynomials. We found

$$m_{avg} = 1210925473348664946846 0.$$

Sieving as in Remark 6.1.2 shows that Polynomial R_1 has a full yield 5.9 times that of the polynomial obtained from m_{avg}.

6.1.5 Some Timing Considerations

It is instructive to examine the comparitive timings of the three searches described here. The actual search for RSA-130 polynomials (that is, the generation of the P_i polynomials) occupied approximately three months on each of four processors [37]. Generation of the polynomials Q_i - using primitive means but with knowledge of what to look for - occupied approximately one month on one processor. Generation of the polynomials R_i - using Procedure 5.1.4, including forcing polynomials to have many projective roots modulo small p - occupied approximately 36 hours on one processor. Hence, even leaving non-projective roots modulo small p^k to chance we obtain an improvement of approximately a factor of two in a trivial amount of time.

6.2 RSA-140

Full details of the RSA-140 factorisation are to be found in [16]. Here of course we comment only on the polynomial selection. Since we used both skewed and non-skewed searches, this is a good chance to compare the results obtained and demonstrate the benefit gained using Procedure 5.1.6 over Procedure 5.1.4.

We need to be mindful now that each "polynomial" is in fact a pair of polynomials. In the case of non-skewed polynomials, this is not confusing because all values of m are similar. However, in this section and the next we compare amongst skewed polynomials, and compare skewed polynomials to non-skewed polynomials. We shall now refer explicitly to polynomial *pairs* when necessary to avoid ambiguity.

6.2.1 Non-skewed Polynomials

For the sake of comparison we include the best non-skewed polynomial pair found in the initial search for RSA-140 polynomials using Procedure 5.1.4. We met this pair earlier in Example 5.1.5. We have

$$\begin{aligned} G_1(x,y) &= 2378666111034213000000\,x^5 \\ &\quad -5148567155828225103040\,x^4y \\ &\quad -4722668925346720843884\,x^3y^2 \\ &\quad +6545365626333869758617\,x^2y^3 \\ &\quad -3356924353646091366162\,xy^4 \\ &\quad -5142225622472630020004\,y^5 \\ \\ G_2(x,y) &= x-6171197423044469387519 13\,y. \end{aligned}$$

with $\alpha(G_1) = -4.2$ and $\chi(G_1) = 0.011$. When considering *skewed* polynomials of course χ becomes an inappropriate quantity to consider, but we deal with this problem in the next subsection.

As was the case with RSA-130, we compare the yield of the best non-skewed polynomial to that of an average non-skewed polynomial. We chose an average non-skewed polynomial for RSA-140 using the same procedure as described at the end of Section 6.1.4 for RSA-130 polynomials. That gave

$$m_{avg} = 4403954599233371015332 11.$$

Polynomial G_1 has a full yield 5.9 times that of the polynomial given by m_{avg}. Notice that this is (perhaps coincidentally) the same improvement over the average case that is noted in Section 6.1.4 for RSA-130.

6.2.2 Skewed Polynomials - Local Considerations

The sieving for RSA-140 was conducted using a combination of sieving techniques. We performed line sieving using the CWI siever, and lattice sieving using the AKL siever (see Section 2.1.3). The final stage of the polynomial selection process is of course to conduct sieving experiments on the chosen few, with the best performing polynomial pair in those experiments becoming the chosen one.

Remark 6.2.1 The sieving experiments for RSA-140 were conducted at CWI using only the CWI siever. The rational factor base bound was 8000000, the algebraic factor base bound 16777215, and the large prime bounds 500000000 and 1000000000 respectively. To obtain a reasonable profile of a polynomial pair over the entire sieve region in a short period of time, we used a sample of b-values across the region rather than every b-value in a short interval. Each pair was sieved over the same *number* of (a, b) pairs in a region skewed appropriately for that polynomial. It is possible that using only a sample of b-values may cause some projective behaviour modulo small p to be over or under emphasized during the experiment. However we do not consider this a major problem.

Although the $\mathbb{E}$ ranking procedure works well, the value $\mathbb{E}(F)$ for any given F has no real physical significance - it's merely the ordering on $\mathbb{E}$ over a set of polynomials that is relevant. In this section and the next we would like to interpret physical differences, inasmuch as they influence yield, between the polynomials under investigation. Hence, we also use data obtained as a first filter on polynomials from Procedure 5.1.6. In particular, we have

$$I(F, S) = \log\left(\sqrt{\iint_S F^2(x, y)\, dxdy}\right),$$

which we use to compute the average size of F over its rectangle S. Recall that S has length to width ratio s. To compare relative sizes of different polynomials we compute $I(F, S)$, with the area of S invariant across the polynomials. We use

$$S = \{(x, y) \in \mathbb{R} : -\sqrt{s} \le x \le \sqrt{s} \text{ and } -1/\sqrt{s} \le y \le 1/\sqrt{s}\}$$

and take $I(F, S)/4$ to be the average log size of F over S. We then use

$$E(F) = I(F, S)/4 + \alpha(F)$$

as an initial and approximate rating of F.

Notice also that construction of the E rating is similar to the ideas underpinning the procedure of [23] mentioned at Remark 6.1.1. We should be wary that, as a ranking mechanism, E is not as reliable as $\mathbb{E}$. Indeed, we do not even bother to consider

$F_2(x, y) = x - my$ in computing $E(F)$. We do find however that, apart from being a useful and quick first filter, E adds to our understanding of the results.

Table 6.7 gives relevant statistics on the top five candidates for RSA-140, according to experiments conducted pursuant to Remark 6.2.1. The sixth polynomial pair in the table, $\overline{F}_{140}$, we describe later.

Poly.	Rel. Yield	Av. Size	α	E	$\mathbb{E}$ rank
A_{140}	1.00	47.91	-7.01	40.82	2
B_{140}	0.965	47.71	-6.57	41.14	1
C_{140}	0.957	48.13	-6.85	41.28	4
D_{140}	0.931	47.95	-6.91	41.04	3
E_{140}	0.930	45.95	-5.00	40.95	5
$\overline{F}_{140}$	0.128	48.97	-0.17	48.80	∞

Table 6.7: Relative yields of the top RSA-140 polynomials

Although the $\mathbb{E}$ rank given in the table refers to the rank revealed by $\mathbb{E}$ over all polynomials generated, the same cannot be said of the E values in the table. Several other polynomials (with similar values of m) had E values similar to those in the table, but were shown by $\mathbb{E}$ and sieving experiments to have inadequate yields.

Polynomial pair A_{140}, which we have seen before, is the one used for the factorisation. Indeed, $A_{140} = (F_1, F_2)$ with

$$
\begin{aligned}
F_1(x, y) \quad = \quad & 439682082840\, x^5 \\
& +390315678538960\, x^4 y \\
& -7387325293892994572\, x^3 y^2 \\
& -19027153243742988714824\, x^2 y^3 \\
& -63441025694464617913930613\, x y^4 \\
& +318553917071474350392223507494\, y^5 \\
\\
F_2(x, y) \quad = \quad & x - 34435657809242536951779007\, y
\end{aligned}
$$

and $s \approx 4000$. Notice that a_5 factors as $2^3 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 41 \cdot 29759$. Since also $4|a_4$ and $2|a_3$, $F_1(x, y)$ is divisible by 8 whenever y is even. $F_1(x, y)$ has at least three roots x/y modulo each prime from 3 to 17 (some of which are projective), and an additional 35 such roots modulo the 18 primes from 19 to 97.

By way of comparison to Figure 6.1 we include a similar figure, Figure 6.2 for this F_1. We use $-44000 \leq x \leq 44000$ and $1 \leq y \leq 22$ to give the region displayed approximately the same area as that in Figure 6.1.

As with R_1 for RSA-130, the most fertile sources of relations are the valleys cut by the real roots eminating from the origin.

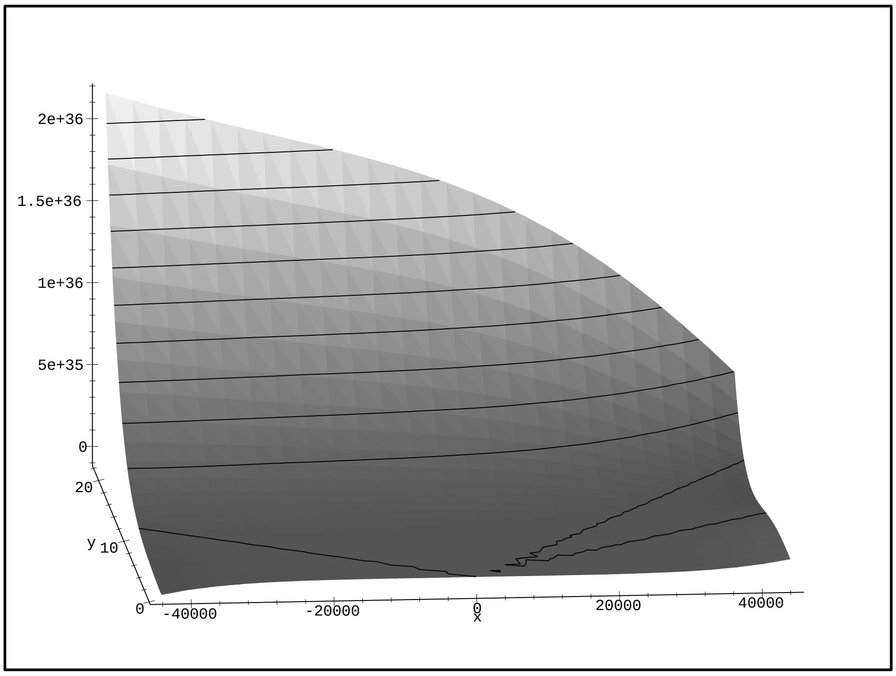

Figure 6.2: $F_1(x, y)$ for RSA-140

We are now in a position to compare the polynomial pair A_{140} to the best non-skewed pair (G_1, G_2) above. We performed sieving experiments of the type described in Remark 6.2.1 on (F_1, F_2) and (G_1, G_2), using $s = 1$ for G_1. We found the yield of the best skewed pair A_{140} is 1.61 times that of (G_1, G_2). From this we also estimate that A_{140} has a yield approximately 9.5 times that of the average non-skewed selection.

We note that the average size of G_1 is 52.27. This is significantly larger than that of the top five polynomials in Table 6.7, and is due to the fact that m is chosen significantly larger in the skewed case to force the leading coefficients of the skewed polynomials to be so small. Hence, particularly when looking close to the origin, considering only the non-linear polynomial in E favours the skewed case.

6.2.3 Skewed Polynomials - Global Considerations

We now compare A_{140} to a skewed selection of average yield. We generated a large random sample of skewed polynomials using Procedure 5.1.6 with randomised a_d in the appropriate range and without rotations which would normally secure good non-projective root properties. We talk more about the *distribution* of random skewed polynomials when discussing RSA-155. For now we compare A_{140} to a particular average selection. We took the average selection to be the polynomial in the sample whose E rating was closest to the mean value. This gives the pair $\overline{F}_{140}$ in Table 6.7.

As shown in Table 6.7, we find the yield of A_{140} is 7.8 times that of the average skewed selection.

Over the entire random sample we found the mean average size of the quintic polynomials F_1 to be 49.15 and the mean $\alpha(F_1)$ to be -0.35. Comparing to the values of the top five candidates in Table 6.7 suggests that most of the benefit we are obtaining comes from root properties rather than size.

6.2.4 Some Timing Considerations

It is estimated that the time spent searching for RSA-140 polynomials, including an initial search for non-skewed polynomials and the developmental phase of the skewed polynomial search, is approximately equivalent to 2000 CPU hours on one 250 MHz Origin 2000 processor. This is very approximately equivalent to 60 MIPS-years. Given that the sieving time was approximately 2000 MIPS-years, we arrive at the question of whether it would have been worthwhile to continue searching for polynomials rather than start sieving.

In the case of RSA-140 there were pragmatic considerations which made it appropriate to stop the polynomial search when we did. We wanted to use increased idle time on workstations over the Christmas period for sieving, so we stopped the polynomial search just before Christmas 1998.

However the question remains. To consider this question we use the larger sample of polynomials examined during the RSA-155 polynomial search.

6.3 RSA-155

First we give the necessary local considerations by examining the top few candidates and their properties, and the performance of the $\mathbb{E}$ ranking. We then move to global considerations. We consider a large sample of randomly generated skewed polynomials and compare the yield of the pair being used for the factorisation to that of a pair from the random sample with average yield. We then turn to the random sample as a whole, and compare its distribution to that of the sample of polynomials generated during the search. Finally, we use the sample of generated polynomials and some approximations to consider the trade-off between polynomial search time and sieving time.

6.3.1 Local Considerations

As in the RSA-140 factorisation, sieving for RSA-155 was conducted using both the AKL and CWI sievers. A large portion of the RSA-140 relations (55%) were generated using the AKL siever. Final statistics are not yet available, but we expect that portion to be larger for RSA-155 (new contributors of sieving machines are using the AKL siever). Hence, sieving experiments on the top few RSA-155 candidate polynomials were conducted on both sievers.

Remark 6.3.1 RSA-155 sieving experiments using the CWI siever were again conducted at CWI in the manner decribed in Remark 6.2.1. Experiments on the AKL siever were run by Arjen Lenstra as follows. For each polynomial pair sieving was conducted over each special q in the ranges $i \cdot 10^7 \le q \le i \cdot 10^7 + 500$ for $i = 2, 3, \dots, 12$. Since the precise number of special q per polynomial is variable, we use the number of relations obtained per special q as the yield measure in these experiments. For implementation specific reasons, the smoothness bounds used on the AKL siever were slightly different to those used on the CWI siever. On the AKL siever, the rational factor base bound was 3497867, the algebraic factor base bound 12174433. The large prime bounds were the same as in Remark 6.2.1.

We note that in addition to actual yield, time per relation is a relevant quantity to compare between polynomials. For polynomials whose yields are very close, the average time per relation may well determine which polynomial is used. This situation has not yet arisen in practice. Moreover, empirically determined time per relation figures can be unreliable since they depend heavily on the load, memory and cache properties of individual machines. Hence we report here only the yield figures.

Table 6.8 gives statistics on the top eight candidates for RSA-155. They are listed in the order revealed by sieving experiments with the AKL siever. Polynomial pair $\overline{F}_{155}$ is referred to later.

Poly.	Rel. Yield (AKL)	Rel. Yield (CWI)	$\mathbb{E}$ rank	Av. Size	α	E
A_{155}	1.00	1.00	1	50.51	−6.25	44.26
B_{155}	0.99	0.96	2	52.43	−6.44	45.99
C_{155}	0.95	0.99	3	51.97	−6.59	45.37
D_{155}	0.89	0.93	4	51.11	−5.74	45.37
E_{155}	0.86	0.85	5	52.08	−6.83	45.22
F_{155}	0.86	0.90	6	52.24	−6.07	46.17
G_{155}	0.85	0.89	8	52.04	−5.43	46.11
H_{155}	0.76	0.83	7	52.95	−6.63	46.32
$\overline{F}_{155}$		0.07	∞	55.69	−0.29	55.40

Table 6.8: Relative yields of the top RSA-155 polynomials

Tests with the CWI siever place C_{155} higher and E_{155} lower than with the AKL siever. There is a strong correlation between the ranking revealed by $\mathbb{E}$ and that of the sieving experiments, particularly with the AKL siever.

As was the case with RSA-140, we comment that the ranking of E values, though informative, may be misleading. Several other polynomials had E ratings as good as the polynomials in the table (with similar m values), but were shown by $\mathbb{E}$ and sieving experiments to have lesser yields.

The pair A_{155}, being used for the factorisation, is

$$\begin{aligned} F_1(x,y) \quad = \quad & 119377138320\, x^5 \\ & -80168937284997582\, x^4 y \\ & -66269852234118574445\, x^3 y^2 \\ & +11816848430079521880356852\, x^2 y^3 \\ & +7459661580071786443919743056\, xy^4 \\ & -40679843542362159361913708405064\, y^5 \end{aligned}$$

$$F_2(x,y) \quad = \quad x - 39123079721168000771313449081\, y$$

with $s \approx 10800$. We have $a_5 = 2^4 \cdot 3^2 \cdot 5 \cdot 11^2 \cdot 19 \cdot 41 \cdot 1759$. Also, $F_1(x,y)$ has 21 roots x/y modulo the six primes from 3 to 17 (some of which are projective), and an additional 34 roots modulo the 18 primes from 19 to 97. Notice that F_1 has root properties just as good as the other polynomials in the table. Compared to the other polynomials in the table however, F_1 has unusually small average size. This is a nice example of root properties and size *combining* to produce our best polynomials.

6.3.2 Global Considerations

A sample of 10000 random skewed polynomials was generated for RSA-155 using the same procedure as for RSA-140. We refer to this sample as the *random sample*. We again chose an average skewed polynomial to be one whose E rating is closest to the mean of the random sample. This gives the pair $\overline{F}_{155}$ of Table 6.8. We find the yield of A_{155} is 13.5 times that of $\overline{F}_{155}$. Comparing this to the figure of 7.8 for the RSA-140 selection we find that the RSA-155 selection is about 1.7 times better, relatively speaking, than the RSA-140 selection.

Over the entire random sample we found the mean average size to be 55.4 and the mean α to be -0.1. Comparing to the values of the top eight candidates in Table 6.8 again suggests that most of the benefit is coming from root properties, although we do have more benefit from size here than was the case for RSA-140.

We generated a large sample of candidate polynomials during the RSA-155 search. As a first filter, we accepted polynomials for which $E(F_1) \leq 47.0$. We found 8200 such polynomials, and these form the *generated sample*. Relative yield is the best measure of the value of the generated sample, but another useful measure is the frequency with which good polynomials occur compared to the random sample.

We examine this by comparing the distribution of the generated polynomials to that of the random sample. Actually we examine the distribution of E values of these polynomials, because the E measure is sufficiently quick to compute for large samples, and it has some physical significance. Hence the term *distribution of good polynomials* refers to the frequency distribution of $E(F_1)$ over the relevant sample.

Figure 6.3 shows distributions of the random and generated samples.

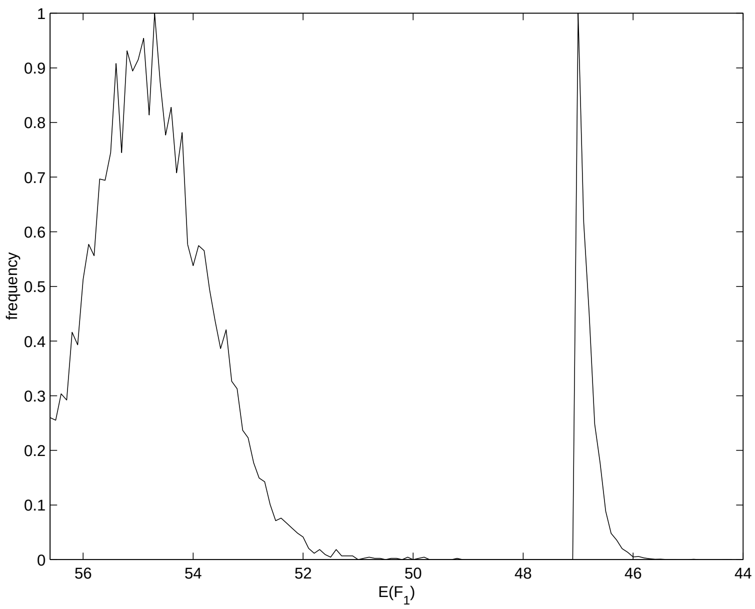

Figure 6.3: Distribution of random and generated polynomials for RSA-155, see Remark 6.3.2

The horizontal axis gives E ratings; better polynomials have lower E ratings and occur towards the right hand side. The vertical axis gives the frequency at which each rating occurs, relative to the modal rating. The leftmost peak is the random sample. We are interested in the right hand tail of this curve. The smallest rating found in the random sample of 10000 polynomials is $E = 49.2$. The largest rating considered in the generated sample is 47.0, so the generated sample lies entirely within the unobservable tail of the random sample.

The rightmost peak is the generated sample.

Remark 6.3.2 The frequencies of the generated sample have been renormalised to the value at $E = 47.0$, so that we may see them.

As we saw in Table 6.8, the best polynomials have approximately $E < 46.0$. These polynomials lie in the unobservable tail of the generated sample, and hence in the unobservable tail of the unobservable tail of the random sample.

For the sake of completeness we include the analogous figure for RSA-140 polynomials (Figure 6.4). The random sample here contains 5700 polynomials. The generated sample contains far fewer polynomials (400), and is not neatly distributed. Again this is because the search procedure was under development during the RSA-140 search. However, it is useful to note that the random sample is distributed similarly to that of RSA-155, and the generated sample lies well into the tail of the random sample.

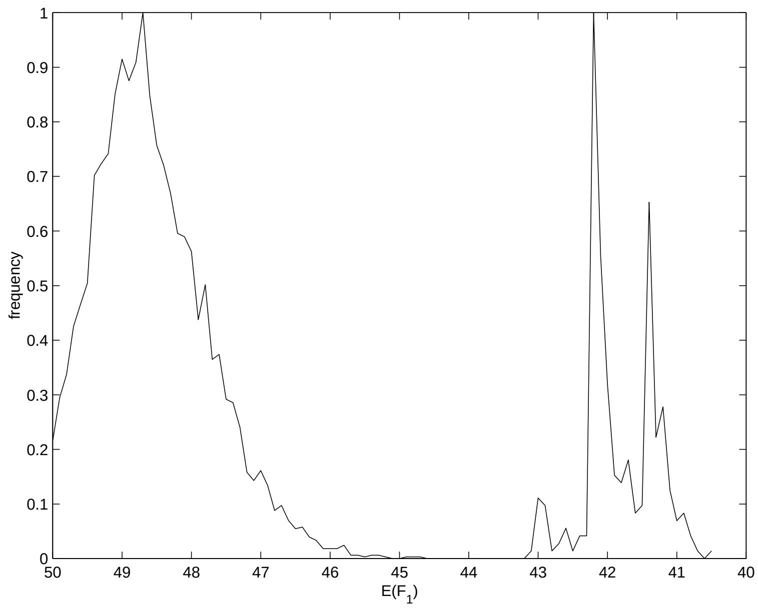

Figure 6.4: Distribution of random and generated polynomials for RSA-140

Let us return to the RSA-155 figure and quantify the extent of the unobservability. We denote by $\mu_r(E)$ the relative frequency in the random sample of polynomials of rating E.

Remark 6.3.3 The distribution shown in Figure 6.3 actually counts E values in intervals of length 0.1. So, formally we regard $\mu(E)$ as being the relative frequency of polynomials F_1 with rating $E - 0.05 \leq E(F_1) \leq E + 0.05$.

The next step is to fit a curve to the $\mu_r(E)$ distribution. Using least squares regression to fit a polynomial to $\log \mu_r(E)$ we found the best fit using

$$\mu_r(E) = \exp(a + bE + cE^2)$$

with

$$a = -1258, \quad b = 45.8, \quad c = -0.417.$$

Figure 6.5 shows the fit of this curve with the random sample.

The quantity

$$\nu_r(E_1, E_2) \;=\; \frac{\mu_r(E_1)}{\mu_r(E_2)} \;=\; \exp\left\{(E_1 - E_2)(b + c(E_1 + E_2))\right\}$$

gives the frequency at which polynomials of rating E_1 appear compared to those with rating E_2. Table 6.9 shows this quantity at some interesting points on the curves in Figure 6.3 (the modal rating in the random sample is 54.6).

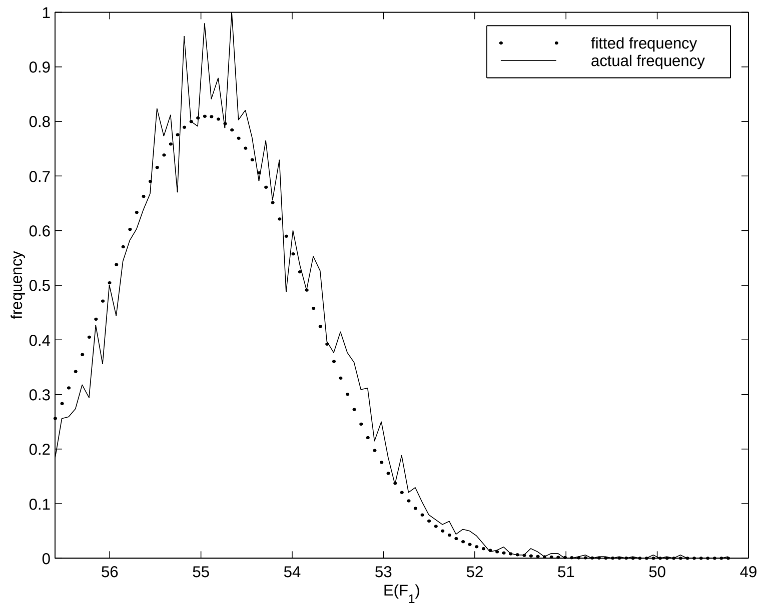

Figure 6.5: Actual and fitted distributions for random sample

Remark 6.3.4 It might be considered useful to examine the cumulative frequency of E ratings. That is, asking "what is the relative frequency of polynomials with ratings at least as good as E?". However since the frequency decreases so quickly as a function of E, the relative frequency itself becomes the determining factor. We content ourselves with examining just the relative frequency in the context of Remark 6.3.3.

Hence, our best few polynomials occur approximately 10^{18} times more rarely than average polynomials. Moreover our best few polynomials occur more than one million times less often than the cut-off polynomials. If we had been searching at random then finding our cut-off polynomials, let alone our best few, would have been out of the question.

6.3.3 Some Timing Considerations

Fortunately, we do not search at random. Our search procedure is of course biased towards finding good polynomials. We now focus on the distribution of the generated sample rather than the random sample, to deal with the question "for how long should we search?".

Similarly to $\mu_r(E)$ above, denote by $\mu_g(E)$ the relative frequency in the generated sample of polynomials with rating E. Using least squares regression to fit a polynomial to $\log \mu_g(E)$ we found the best fit using

$$\mu_g(E) = \exp(a + bE)$$

E_1	E_2	$\nu(E_1, E_2)$
54.6	47.0	$10^{11.5}$
54.6	45.0	$10^{18.0}$
47.0	46.0	$10^{3.1}$
47.0	45.0	$10^{6.5}$
47.0	44.0	$10^{10.3}$

Table 6.9: Relative frequencies of good polynomials

with

$$a = -278.5, \quad b = 5.92.$$

Notice that we obtain a linear exponential for the generated sample as opposed to a quadratic exponential for the random sample. Figure 6.6 shows the fit.

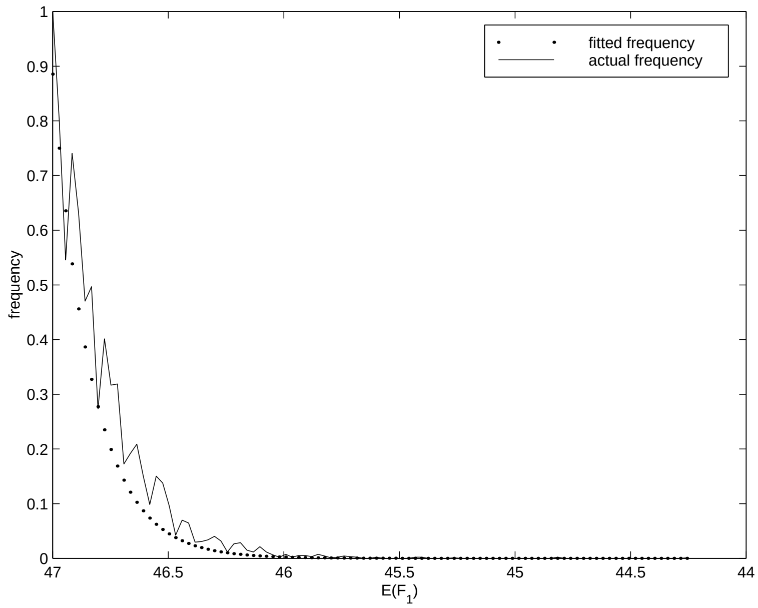

Figure 6.6: Actual and fitted distributions for generated sample

Let

$$\nu_g(E_1, E_2) = \frac{\mu_g(E_1)}{\mu_g(E_2)} = \exp\{b(E_1 - E_2)\}$$

be the relative frequency at which polynomials with rating E_1 appear compared to those with rating E_2, in the generated sample. We use $\nu_g(E_1, E_2)$ below to estimate the E ratings of polynomials we might expect to obtain from a given amount of search effort.

We will combine such an estimate with an approximation of the expected change in yield. To quantify the expected change in yield as a function of E we extrapolate crudely from pairs A_{155} to H_{155} in Table 6.8. Assume that, at least locally, yield changes approximately linearly with E. Clearly this is not true, yield does not even change monotonically with E, but this should suffice to give a rule-of-thumb approximation. Using the yield figures from the AKL siever, we find that every decrease of 0.1 in E corresponds crudely to an increase of 1.2% in yield. Notice that this is the same approximation obtained from averaging between A_{155} and $\overline{F}_{155}$.

Final statistics on the actual sieving time for RSA-155 are not yet available. We suggest a reasonable advance estimate is 8000 MIPS-years. This is derived using the L-function to extrapolate from the RSA-140 sieving time (see Section 2.1.4), and factoring in the better polynomial selection for RSA-155 relative to RSA-140. So the estimate is $2000 \cdot 7 \cdot 7.8/13.5 \approx 8000$ MIPS-years.

That is, every 1% improvement in polynomial yield saves 80 MIPS-years in sieving time. Table 6.10 shows the expected benefit obtained from κ times the search effort we actually invested, for some useful κ. The second column uses $\nu_g(E_1, E_2)$ to estimate the expected change in E as a result of the κ-altered search effort, the third column uses the above rule-of-thumb to estimate the corresponding change in yield, compared to A_{155}. The final two columns give the expected change in polynomial search time and the expected change in sieving time, respectively, in MIPS-years.

κ	E	Yield (%)	Search Time (MY)	Sieve Time (MY)
0.1	-0.39	-4.7	-90	$+380$
0.2	-0.27	-3.2	-80	$+260$
0.5	-0.12	-1.4	-50	$+110$
1	-	-	-	-
2	$+0.12$	$+1.4$	$+100$	-110
5	$+0.27$	$+3.2$	$+400$	-260
10	$+0.39$	$+4.7$	$+900$	-380
15	$+0.46$	$+5.4$	$+1400$	-430

Table 6.10: Costs and benefits of polynomial search time

The point to stop searching for polynomials is the point at which the marginal cost exceeds the marginal benefit. That is, at approximately twice the effort we invested for the RSA-155 search. We used approximately twelve machines for the RSA-155 search. Given that we have well over 200 machines available for sieving, it would be no practical difficulty to use, say, 25 machines for the polynomial search over the same period of time as we used for the actual search.

We caution against over-reliance on the actual figures in Table 6.10. We suspect μ_g is overly pessimistic, and of course the rule-of-thumb for yield as a function of E is only approximate. Still, it does seem reasonable to conclude that, despite the benefit not being great in absolute terms, it could have been worthwhile using up to twice the effort invested in the RSA-155 polynomial search.

6.4 Summary

In this chapter we have examined two new factorisation records, RSA-140 and RSA-155, and one old one, RSA-130.

6.4.1 RSA-130

We re-examined the polynomial selection task for RSA-130 as a means of testing Procedure 5.1.4 and the $\mathbb{E}$ rating. After testing on several sets of polynomials we conclude that $\mathbb{E}$ gives a reliable pre-sieving ranking of yield.

Using the $\mathbb{E}$ rating and Procedure 5.1.4 we found, in a tiny fraction of the time spent on the actual RSA-130 polynomial search, several significantly better polynomials. Our best RSA-130 polynomial has a full yield twice that of the polynomial used for the factorisation, and approximately 5.9 times that of a non-skewed polynomial of average yield.

In essence, the RSA-130 results begin to demonstrate the benefit of knowing "what to look for".

6.4.2 RSA-140

The RSA-140 and RSA-155 results demonstrate the benefit of also knowing "how to look for it".

The RSA-140 factorisation is the first major test of Procedure 5.1.6. Our best polynomial pair, used for the factorisation, has a full yield close to eight times that of a skewed pair of average yield. Approximately a factor of four in that eight comes from root properties, approximately a factor of two from size. Better polynomials could have been obtained, but the search was truncated for practical reasons.

For comparison, we also searched initially for non-skewed polynomials using Procedure 5.1.4. The best non-skewed polynomial found has a full yield approximately 5.9 times that of an average non-skewed polynomial.

6.4.3 RSA-155

Locally, the RSA-155 results exemplify the benefit obtain by finding polynomials with good combinations of size and root properties. The best polynomial pair so found has a full yield approximately 13.5 times better than an average skewed selection.

Globally, we find that such polynomials occur approximately 10^{18} times less often in a random skewed sample, than average polynomials. The sample of polynomials generated during the search however, is much more favourably distributed. Using this distribution and some further approximations, we estimate it may have been beneficial to invest up to twice the effort that went into polynomial selection for RSA-155. The expected gain in yield over the polynomial pair used is not great, but it does exceed the cost of obtaining it.

In any event, it is reasonable to conclude that for large RSA factorisations our methods are able to find polynomial pairs whose yields are 10–15 times greater than the average selection. This makes the sieving task for factorisation of 512 bit RSA moduli entirely do-able with a small collection of machines. Indeed, it has been done.

Chapter 7

Conclusions and Further Work

In this chapter we summarise the conclusions of this thesis, and suggest some areas for further research.

7.1 Conclusions

Detailed conclusions are given at the end of Chapters 3–6. In this section we merely summarise what is said there.

- Good number field sieve polynomials are polynomials which have good yield. Yield can be adequately accounted for by combining measures of size and root properties.

- Once yield is correctly accounted for, polynomials with good yield must be found. We improve on previous efforts by introducing new techniques for finding base-m polynomials with good combinations of size and root properties. These techniques work best, particularly with regard to non-projective root properties, when the non-linear polynomial is highly skewed. Under conditions experienced in large RSA factorisations, we are able to exploit root properties alone to increase yield by up to a factor of four.

- Using our techniques for N in the current range of interest it is cost effective to find polynomials with yields 10–15 times better than a random selection. We factorised RSA-140 using a polynomial which is almost that good. We are factorising RSA-155 using a polynomial which is that good.

- 512 bit RSA moduli are demonstrably insecure.

7.2 Further Work

We suggest the following areas of further research.

- There may be implementation specific reasons for users to prefer non-skewed polynomials. It should be possible to introduce a sieve-like procedure to identify non-skewed F_1 with good non-projective root properties. This may not be as succesful as for highly skewed F_1, since we do not have the freedom of inspecting many rotations for each possible F_1. Even if we can find them, we wouldn't expect non-skewed polynomials with excellent non-projective root properties to have significantly better yields than the skewed polynomials we already find.

- As we move to higher degree F_1, higher degree rotations could be considered. Quadratic rotations would be appropriate for sextic F_1.

- We might consider applying our improvements to the selection of base-m_1, m_2 polynomials (Remark 2.3.2). We might also consider extensions of Montgomery's Two Quadratics method (Section 2.3.1) and novel methods for new degree pairs (Remark 3.1.1).

- Since we now have a better understanding of the generation of good polynomials it may be time to reconsider multiple polynomial versions of the number field sieve. When using several non-linear polynomials, the proximity of regions of maximal yield (usually, real roots) should be considered (see Remarks 2.3.1 and 4.2.1).

  One consequence of using rotations in polynomial selection is that several base-m polynomials F_1 can be found with the same common root m. It could be worthwhile to consider using several such polynomials $F_{1,j}$ and seeking relations between each $F_{1,j}$ and F_2. Perhaps with sufficiently many good $F_{1,j}$, only the regions of maximal yield need be considered. The obvious disadvantage of such a scheme is that the matrix size increases linearly with the number of polynomials.

- We should apply our techniques to discrete logarithm number field sieve computations.

- Having at least partially addressed the polynomial selection problem, it now becomes even more crucial that we improve the matrix reduction step. This is also relevant to discrete logarithm computations. The promising avenues for improvement are better filtering strategies and parallelisation of the reduction code.

- Factorisation of smaller RSA moduli, like RSA-150, would be useful to give a more complete picture of the growth of actual factorisation effort with N. Factorisation of larger RSA moduli, as well as being useful, would be exciting. Hence, we should factorise more RSA moduli.

Appendix A

Appendix to Chapter 4

Polynomials $A, \ldots, K$ are listed below. The values of m given are $m \in \mathbb{Z}$ for which $f(m) \equiv 0 \bmod N$. The values of N are C106 for polynomials $A, \ldots, E$ and polynomials H and I, C105 for polynomials F and G, and C107 for polynomials J and K.

Polynomial A:

$$\begin{aligned}&106422971201966162010185797481984649946687+\\&15716891810512433152501163 7x - 323379595900x^2\end{aligned}$$

$$\begin{aligned}m = &31181176714425679596439277079929546857772784928744 1\backslash\\&417195888224875673003757757525998997704760967662422630\end{aligned}$$

Polynomial B:

$$\begin{aligned}&-58535465962950604788770735849031669686845+\\&578123152107916050639034324x + 660940091871x^2\end{aligned}$$

$$\begin{aligned}m = &111266350151832591590373321222840072472133768682060\backslash\\&5812518391957850167078163045569883641392384840611818322\end{aligned}$$

Polynomial C:

$$\begin{aligned}&-80444723076532128931843884067440931877697+\\&671898769354767184209613115x + 876541800001x^2\end{aligned}$$

$$\begin{aligned}m = &644385945238412299450097726772298730429521837407426\backslash\\&656132710287589175267555416671359532826085727240133210\end{aligned}$$

Polynomial D:

$$-4560132934901424596132446855946800312514 3+$$
$$40586388695680988961101222 0x+875883403741x^2$$

$$m=57022157889652460507276414622928637851608638531004\backslash$$
$$7513013419381527088912105584724979693796690373689178237$$

Polynomial E:

$$-43070512279968963999727149653384015128406$$
$$-14064499759408820601443835 3x+274174364727x^2$$

$$m=21431385359461632490985189041791385017574508889045\backslash$$
$$6629204834574379795020566498337694386071915713661516800$$

Polynomial F:

$$540759062604782971357139536186424874771+$$
$$868170693335194654836416 12x+342910527737x^2$$

$$m=22914359055586946906211501353855768192316423575426\backslash$$
$$6217765793563500275674926893987223245481401160544005942$$

Polynomial G:

$$1291287673000652336311682295362679824208 00$$
$$-91304927318176881696255321 8x+1242060255079x^2$$

$$m=22914359055586946906211501353855768192316423575426\backslash$$
$$6217765793563500275674926893987223245481401160544005942$$

Polynomial H:

$$-3243028756049597614391031715982337625514 4$$
$$-10164316373443673606696029 4x+190030476113x^2$$

$$m=17900441287572625768481534121337659378990978888143\backslash$$
$$77815816769105476827696665209945565825606429787588581699$$

Polynomial I:

$$16408608000145603417923876654325668771382 7$$
$$-40196864605174227034428017 2x-785083260639x^2$$

$$m=17900441287572625768481534121337659378990978888143\backslash$$
$$77815816769105476827696665209945565825606429787588581699$$

Polynomial J:

$$-31165399435941867031977533013643451350698 6+$$
$$763119703166287854853198889x - 241799514805x^2$$

$$m = 12637530599467776761853128412624277137347729851839\backslash$$
$$92404839228760524925327079726440981323065372540515548489 2$$

Polynomial K:

$$-46786964108579179806101863478910720071558+$$
$$-425704283028714253779269315x - 540161776283x^2$$

$$m = 12637530599467776761853128412624277137347729851839\backslash$$
$$924048392287605249253270797264409813230653725405155484892$$

Appendix B

Appendix to Chapter 6

Below are polynomials referred to in Chapter 6. Non-skewed polynomials are defined uniquely by m (see Section 2.3), so we give only m for these polynomials. For skewed polynomials, we give F_1, F_2 and the skewness s.

B.1 RSA-130 Polynomials

Table B.1 gives values of m for polynomials R_i, $i = 1, \ldots, 5$.

Poly	m
1	12429620102099690356862
2	12429620102099690356861
3	12429620102099690356863
4	13451029676646753000757
5	12400786914908592973618
6	12664454168907537623814

Table B.1: Values of m for R_i

Table B.2 contains the values of m for polynomials $P_1, \ldots, P_{15}$ (provided by Arjen Lenstra) and $Q_1, \ldots, Q_{18}$.

$P_1, \ldots, P_{15}$		$Q_1, \ldots, Q_{18}$	
P_i	m	Q_i	m
1	10519776768693341771145	1	13892376347633905755115
2	12112464325781598662255	2	12453346471414472759941
3	12175183789358781924382	3	12189668945503746069685
4	12922982589397980905651	4	11837358189073863960965
5	10056778742160802578928	5	14227836633450858685725
6	12893568754859383127665	6	12846317334855496412374
7	13239320351370744041131	7	12485318267855789022719
8	12506435569527239916746	8	13664023713239125138661
9	12666132133378233425814	9	14262547698921937056113
10	10844346817052874470999	10	13755004021960592085464
11	13139341559800540682218	11	14214149085376118983291
12	12857394860965184611325	12	15151852662623823374781
13	11856745579968929283390	13	14185394352093247029946
14	12574411168418005980468	14	12351139031991610954191
15	11507478393662235457656	15	14601881988167170300659
		16	13603479675779569518553
		17	13809622636367237837331
		18	12464197256082744853511

Table B.2: Values of m for P_i and Q_i

B.2 RSA-140 Polynomials

A_{140}:

$$
\begin{aligned}
F_1(x,y) &= \quad 439682082840\, x^5 \\
&\quad +390315678538960\, x^4 y \\
&\quad -7387325293892994572\, x^3 y^2 \\
&\quad -19027153243742988714824\, x^2 y^3 \\
&\quad -63441025694464617913930613\, x y^4 \\
&\quad +318553917071474350392223507494\, y^5 \\
\\
F_2(x,y) &= x - 34435657809242536951779007\, y \\
\\
s &= 4096
\end{aligned}
$$

B_{140}:

$$\begin{aligned}
F_1(x,y) &= 475678803600\,x^5 \\
&\quad +12310512454193580\,x^4y \\
&\quad -471955228682812456 22\,x^3y^2 \\
&\quad -188753744778883172 30356\,x^2y^3 \\
&\quad +70859290510917128272 5988833\,xy^4 \\
&\quad -76237857487252581793 2463490775\,y^5
\end{aligned}$$

$$F_2(x,y) = x - 338979455148692720709 38702\,y$$

$$s = 3680$$

C_{140}:

$$\begin{aligned}
F_1(x,y) &= 473378805900\,x^5 \\
&\quad +6786847212725992\,x^4y \\
&\quad -107779980090539302193\,x^3y^2 \\
&\quad -3260181992508395878136 47\,x^2y^3 \\
&\quad +2303400508103580132807 667310\,xy^4 \\
&\quad -1306686150190334964106 092161208\,y^5
\end{aligned}$$

$$F_2(x,y) = x - 55773850015391247110492107\,y$$

$$s = 6200$$

D_{140}:

$$\begin{aligned}
F_1(x,y) &= 569366998200\,x^5 \\
&\quad +27579278413218810\,x^4y \\
&\quad -579998372934903230 01\,x^3y^2 \\
&\quad -494560012317526613653093\,x^2y^3 \\
&\quad +111802304474201423600 5014576\,xy^4 \\
&\quad -981338508886515998832 45735012\,y^5
\end{aligned}$$

$$F_2(x,y) = x - 37563294757862265713468083\,y$$

$$s = 4119$$

E_{140}:

$$\begin{aligned}
F_1(x,y) &= 54960260355\,x^5 \\
&\quad +97578919634740\,x^4y \\
&\quad -3693662646946497286\,x^3y^2 \\
&\quad -190271532437429887148 24\,x^2y^3 \\
&\quad -126882051388929235827861226\,xy^4 \\
&\quad +127421566828589740156 8894029976\,y^5
\end{aligned}$$

$$F_2(x,y) = x - 68871315618485073903558014\,y$$

$$s = 7360$$

$\overline{F}_{140}$:

$$\begin{aligned}
F_1(x,y) &= \quad 9187603793796\,x^5 \\
&\quad +12386461804765297\,x^4y \\
&\quad +469987288306604686609\,x^3y^2 \\
&\quad -889049056208116896399\,x^2y^3 \\
&\quad +1398744126837196850050 0939\,xy^4 \\
&\quad -29615702384694218895 2843\,y^5 \\
F_2(x,y) &= x - 18749758811416934921816359\,y \\
s &= 300
\end{aligned}$$

B.3 RSA-155 Polynomials

A_{155}:

$$\begin{aligned}
F_1(x,y) &= \quad 119377138320\,x^5 \\
&\quad -80168937284997582\,x^4y \\
&\quad -66269852234118574445\,x^3y^2 \\
&\quad +11816848430079521880356852\,x^2y^3 \\
&\quad +7459661580071786443919743056\,xy^4 \\
&\quad -40679843542362159361913708405064\,y^5 \\
F_2(x,y) &= x - 39123079721168000771313449081\,y \\
s &= 10770
\end{aligned}$$

B_{155}:

$$\begin{aligned}
F_1(x,y) &= \quad 9734331382020\,x^5 \\
&\quad +186548816004600576\,x^4y \\
&\quad -2621958757709806297705\,x^3y^2 \\
&\quad -11937100897656690036171818\,x^2y^3 \\
&\quad +686144075682507925299871832 15\,xy^4 \\
&\quad +72327510316160055608800665174636\,y^5 \\
F_2(x,y) &= x - 18636400766678583399319133866\,y \\
s &= 6354
\end{aligned}$$

C_{155}:

$$\begin{aligned} F_1(x,y) &= 1290313469760\,x^5 \\ &\quad +265878007916683818\,x^4y \\ &\quad -79839840387378796571 5\,x^3y^2 \\ &\quad -6913919978250014083817403 0\,x^2y^3 \\ &\quad +381518658826906113738382758 00\,xy^4 \\ &\quad +301077153817651006526347389706989 7\,y^5 \end{aligned}$$

$$F_2(x,y) = x - 2430402600327742999575555144 0\,y$$

$$s = 13103$$

D_{155}:

$$\begin{aligned} F_1(x,y) &= 13773893580720\,x^5 \\ &\quad +293273156850908000\,x^4y \\ &\quad -1262097631040259345842\,x^3y^2 \\ &\quad -7076664823854260804438715\,x^2y^3 \\ &\quad +15958633172059160822941381252\,xy^4 \\ &\quad -60529194441853543661902699832835\,y^5 \end{aligned}$$

$$F_2(x,y) = x - 151358188987776755880994202 98\,y$$

$$s = 5430$$

E_{155}:

$$\begin{aligned} F_1(x,y) &= 2697367246860\,x^5 \\ &\quad +105115408896978962\,x^4y \\ &\quad +3195116446280929272587\,x^3y^2 \\ &\quad -2447107130899453676010244 8\,x^2y^3 \\ &\quad -410312201224383538645857505823\,xy^4 \\ &\quad +29876530689458684852162460785950\,y^5 \end{aligned}$$

$$F_2(x,y) = x - 2767204464562081315011235692 6\,y$$

$$s = 11687$$

F_{155}:

$$\begin{aligned} F_1(x,y) &= 13773893580720\,x^5 \\ &\quad -238707428451700000\,x^4y \\ &\quad -3743293864033106325842\,x^3y^2 \\ &\quad +3422343839391753504266849 5\,x^2y^3 \\ &\quad +219972273530847363657409036632\,xy^4 \\ &\quad -2120792922120149563797447040262880\,y^5 \end{aligned}$$

$$F_2(x,y) = x - 1513581889877767558809942490 3\,y$$

$$s = 10860$$

G_{155}:

$$\begin{aligned}
F_1(x,y) \quad = \quad & 10087787167920\,x^5 \\
& -245018020007667129\,x^4y \\
& -3601309509661665837217\,x^3y^2 \\
& +1198690603804512576976217 3\,x^2y^3 \\
& +25253316913997385921187788966 8\,xy^4 \\
& +6960414190852779014696363651642 52\,y^5
\end{aligned}$$

$$F_2(x,y) = x - 1610861067127907507403226069 1\,y$$

$$s = 9741$$

H_{155}:

$$\begin{aligned}
F_1(x,y) \quad = \quad & 8648697934800\,x^5 \\
& +800666437942682720\,x^4y \\
& -3757414786445679414797\,x^3y^2 \\
& -11483097947130398156334363 3\,x^2y^3 \\
& +96691565654522380316377089613\,xy^4 \\
& -2081067100601209101363405989002 23\,y^5
\end{aligned}$$

$$F_2(x,y) = x - 1661219886953234542299300484 0\,y$$

$$s = 10941$$

$\overline{F}_{155}$:

$$\begin{aligned}
F_1(x,y) \quad = \quad & 453631755\,x^5 \\
& +7707423309885\,x^4y \\
& +5749800439136769185023 17\,x^3y^2 \\
& +1438679581208554647120 54\,x^2y^3 \\
& -55372733115530332680409180475 6\,xy^4 \\
& +4041646265258084586097204313 7\,y^5
\end{aligned}$$

$$F_2(x,y) = x - 1192549947731541967713150737 86\,y$$

$$s = 1080$$

Bibliography

[1] L M Adleman, "Factoring Numbers Using Singular Integers", *Proc. 23rd Annual ACM Symp. on Theory of Computing (STOC)* (1991) pp 64–71.

[2] D Atkins, M Graff, A K Lenstra, and P C Leyland, "The Magic Words are Squeamish Ossifrage", *Advances in Cryptology - ASIACRYPT 94, LNCS* **917** (1995) pp 263–277.

[3] E Bach and R Peralta, "Asymptotic Semismoothness Probabilities", *Math. Comp.* **65** (1996) pp 1717–1735.

[4] D Bernstein, "Bounding Smooth Integers", *Algorithmic Number Theory - ANTS III, LNCS* **1423** (1998) pp 128–131.

[5] D J Bernstein and A K Lenstra, "A General Number Field Sieve Implementation", *The Development of the Number Field Sieve, LNM* **1554** (1993) pp 103–125.

[6] H Boender, *Factoring Large Integers with the Quadratic Sieve*, PhD Thesis, University of Leiden, 1997.

[7] H Boender and H J J te Riele, "Factoring Integers with Large Prime Variations of the Quadratic Sieve", *Experimental Mathematics* **5**(4) (1996) pp 257–273.

[8] Z I Borevich and I R Shafarevich, *Number Theory*, Academic Press, New York, 1966.

[9] R P Brent, "Factorization of the Tenth Fermat Number", *Math. Comp.* **68** (1999) pp 429–451.

[10] R P Brent, "Some Integer Factorization Algorithms using Elliptic Curves", *Aus. Comp. Sci. Communications* **8** (1986) pp 149–163.

[11] D M Bressoud, *Factorization and Primality Testing*, Springer-Verlag, New York, 1989.

[12] J Buchmann, J Loho and J Zayer, "An Implementation of the Number Field Sieve", *Advances in Cryptology - CRYPTO 93, LNCS* **773** (1993) pp 159–165.

[13] D A Buell, *Binary Quadratic Forms, Classical Theory and Modern Computations*, Springer-Verlag, New York, 1989.

[14] J P Buhler, H W Lenstra Jr and C Pomerance, "Factoring Integers with the Number Field Sieve", *The Development of the Number Field Sieve, LNM* **1554** (1993) pp 50–94.

[15] S Cavallar, "Strategies for Filtering in the Number Field Sieve", in preparation (1999).

[16] S Cavallar, B Dodson, A K Lenstra, P Leyland, W Lioen, P L Montgomery, B Murphy, H te Riele and P Zimmerman, "Factorization of RSA-140 using the Number Field Sieve", submitted (1999).

[17] S Cavallar, B Dodson, A K Lenstra, P Leyland, W Lioen, P L Montgomery, H te Riele and P Zimmerman, "211-digit SNFS Factorisation", *Number Theory List Server, 25 April* (1999). Available at `ftp://ftp.cwi/nl/pub/herman/NFSrecords/SNFS-211`.

[18] H Cohen, *A Course in Computational Algebraic Number Theory*, Springer-Verlag, Berlin, 1993.

[19] S D Cohen, "The Distribution of the Galois Groups of Integral Polynomials", *Illinois J. of Math.* **23** (1979) pp 135–152.

[20] D Coppersmith, "Modifications to the Number Field Sieve", *J. Cryptology* **6** (1993) pp 169–180.

[21] D Coppersmith, A Odlyzko and R Schroeppel, "Discrete Logarithms in $GF(p)$", *Algorithmica* **1** (1986) pp 1–15.

[22] J-M Couveignes, "Computing a Square Root for the Number Field Sieve", *The Development of the Number Field Sieve, LNM* **1554** (1993) pp 95–102.

[23] J Cowie, B Dodson, R M Elkenbracht-Huizing, A K Lenstra, P Montgomery and J Zayer, "A World Wide Number Field Sieve Factoring Record: On to 512 Bits", *Advances in Cryptology - ASIACRYPT 1996, LNCS* **1163** (1997) pp 382–394.

[24] S Davis, W Duke and X Sun, "Probabilistic Galois Theory of Reciprocal Polynomials", *Expositiones Mathematicae* **16** (1998) pp 263–270.

[25] T F Denny, B Dodson, A K Lenstra and M Manasse, "On the Factorisation of RSA-120", *Advances in Cryptology - CRYPTO 93, LNCS* **773** (1994) pp 166–174.

[26] T F Denny and V Muller, "On the Reduction of Composed Relations from the Number Field Sieve", *Algorithmic Number Theory - ANTS II, LNCS* **1122** (1996) pp 79–92.

[27] K Dickman, "On the Frequency of Numbers Containing Prime Factors of a Certain Relative Magnitude", *Ark. Mat., Astronomi och Fysik* **22A** 10 (1930) pp 1–14.

[28] W Diffie and M Hellman, "New Directions in Cryptography", *IEEE Trans. Information Theory* **22** (1976) pp 472–492.

[29] B Dixon and A K Lenstra, "Factoring Using SIMD Sieves", Advances in Cryptology - EUROCRYPT 93, LNCS **765** (1993) pp 28–39.

[30] B Dodson and A K Lenstra, "NFS with four large primes: An Explosive Experiment", *Advances in Cryptology - CRYPTO 95, LNCS* **963** (1995) pp 372–385.

[31] T ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *IEEE Trans. Information Theory* **31** (1976) pp 469–472.

[32] M Elkenbracht-Huizing, "A Multiple Polynomial General Number Field Sieve", *Algorithmic Number Theory - ANTS II, LNCS* **1122** (1996) pp 99–114.

[33] M Elkenbracht-Huizing, "An Implementation of the Number Field Sieve", *Experimental Mathematics* **5**(3) (1996) pp 231–253.

[34] P X Gallagher, "The Large Sieve and Probabilistic Galois Theory", *Analytic Number Theory: Proc. Symp. in Pure Math.* **24** (1973) pp 91–101.

[35] R A Golliver, A K Lenstra and K S McCurley, "Lattice Sieving and Trial Division", *Algorithmic Number Theory - ANTS 94, LNCS* **877** (1994) pp 18–27.

[36] D Gordon, "Discrete Logarithms in $GF(p)$ using the Number Field Sieve", *SIAM J. Disc. Math.* **6**(1) (1993) pp 124–138.

[37] S Huddleston, Personal Communication, June 1998.

[38] N Koblitz, "Elliptic Curve Cryptosystems", *Math. Comp.* **48** (1987) pp 203–209.

[39] D E Knuth, "Seminumerical Algorithms: Volume 2" of *The Art of Computer Programming* (3rd ed), Addison Wesley, 1998.

[40] D E Knuth and L T Pardo, "Analysis of a Simple Factorization Algorithm", *Theor. Comp. Sci.* **3** (1976) pp 321–348.

[41] W Kordeki, "On the Connectedness of Random Hypergraphs", *Comment. Math. Prace. Mat.* **25** (1985), no. 2, pp 265–283.

[42] I N Kovalenko, "On the Theory of Random Graphs", *Kibernetika* **4** (1971) pp 1–4 (in Russian).

[43] M Kraitchik, *Theorie des Nombres, Tome II*, Gauthiers-Villars, Paris (1926) pp 195–208.

[44] B A LaMacchia and A M Odlyzko, "Solving Large Sparse Systems over Finite Fields", *Advances in Cryptology - CRYPTO 90, LNCS* **537** (1991) pp 109–133.

[45] R Lambert, *Computational Apsects of Discrete Logarithms*, PhD Thesis, Univeristy of Waterloo, 1996.

[46] A K Lenstra, H W Lenstra and L Lovasz, "Factoring Polynomials with Rational Coefficients", *Math. Ann* **261** (1982) pp 515–534.

[47] A K Lenstra, H W Lenstra Jr, M S Manasse and J M Pollard, "The Factorization of the Ninth Fermat Number", *Math. Comp.* **61** (1993) pp 319–349.

[48] A K Lenstra, H W Lenstra Jr, M S Manasse and J M Pollard, "The Number Field Sieve", *The Development of the Number Field Sieve, LNM* **1554** (1993) pp 11–42.

[49] A K Lenstra and M S Manasse, "Factoring with Two Large Primes", *Math. Comp.* **63** (1994) pp 785–798.

[50] H W Lenstra Jr, "Factoring Integers with Elliptic Curves", *Ann. of Math.* **126** (1987) pp 649–673.

[51] P L Montgomery, "Square Roots of Products of Algebraic Numbers", *Mathematics of Computation 1943 – 1993: Proc. Symp. in Appl. Math* **48** (1994) pp 567–571.

[52] P L Montgomery, "A Block Lanczos Algorithm for Finding Dependencies over $GF(2)$", *Advances in Cryptology - EUROCRYPT 95, LNCS* **921** (1995) pp 106–120.

[53] P L Montgomery, "Small Geometric Progressions Modulo n", manuscript (1995).

[54] P L Montgomery and B Murphy, "Improved Polynomial Selection for the Number Field Sieve", *Extended Abstract, Invited Talk, The Mathematics of Public Key Cryptography Conference*, Fields Institute, Toronto, June 1999.

[55] F Morain, "Analysing PMPQS", manuscript (1993).

[56] B Murphy and R P Brent, "On Quadratic Polynomials for the Number Field Sieve", *Computing Theory, ACSC* **20**(3) (1998) pp 199–213.

[57] B Murphy, "Modelling the Yield of Number Field Sieve Polynomials", *Algorithmic Number Theory - ANTS III, LNCS* **1443** (1998) pp 137–150.

[58] K K Norton, "Numbers with small prime factors, and the least k-th power non-residue", *Memoirs Amer. Math. Soc.* **106** (1971) pp 1–106.

[59] P Nguyen, "A Montgomery-Like Root for the Number Field Sieve", *Algorithmic Number Theory - ANTS III, LNCS* **1443** (1998) pp 151–168.

[60] E Palmer, *Graphical Evolution: An Introduction to the Theory of Random Graphs*, John Wiley and Sons, New York, 1985.

[61] J Pollard, "Theorems on Factorisation and Primality Testing", *Proc. Cambridge Phil. Soc* **76** (1974) pp 521–528.

[62] J Pollard, "A Monte-Carlo Method for Factorization", *BIT* **15** (1975) pp 331–334.

[63] J Pollard, "Factoring with Cubic Integers", *The Development of the Number Field Sieve, LNM* **1554** (1993) pp 4–10.

[64] J M Pollard, "The Lattice Sieve", *The Development of the Number Field Sieve, LNM* **1554** (1993) pp 43–49.

[65] C Pomerance, "The Quadratic Sieve Factoring Algorithm", *Advances in Cryptology - EUROCRYPT 84, LNCS* **209** (1985) pp 169–182.

[66] M O Rabin, "Digital Signatures and Public-Key Functions as Intractible as Factorisation", *MIT Laboratory for Computer Science Technical Report*, MIT/LCS/TR-212 (1979).

[67] H Riesel, *Prime Numbers and Computer Methods for Factorization*, second edition, Birkhauser, Boston, 1994.

[68] R L Rivest, A Shamir and L Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM*, **21(2)** (1978) pp 120–126.

[69] RSA Laboratories. For information on the RSA Challenge see `http://www.rsa.com/rsalabs/html/challenges.html`.

[70] A Shamir, "Factoring Large Numbers with the TWINKLE Device", Extended Abstract (1999).

[71] O Schirokauer, "Discrete Logarithms and Local Units", *Phil. Trans. R. Soc. Lond. A* **345** (1993) pp 409–423.

[72] O Schirokauer, D Weber and T Denny, "Discrete Logarithms: The Effectiveness of the Index Calculus Method", *Algorithmic Number Theory - ANTS II, LNCS* **1122** (1996) pp 337–361.

[73] B Schneier, *Applied Cryptography*, John Wiley & Sons, New York, 1996.

[74] R D Silverman, "The Multiple Polynomial Quadratic Sieve", *Math. Comp.* **48** (1987) pp 329–339.

[75] I N Stewart and D O Tall, *Algebraic Number Theory*, Chapman and Hall, London, 1979.

[76] N Tzanakis and B M M de Weger, "How to Explicitly Solve a Thue-Mahler Equation", *Compositio Math.* **84** (1992) pp 223–288.

[77] J van de Lune and E Wattel, "On the Numerical Solution of a Differential-Difference Equation Arising in Analytic Number Theory", *Math. Comp.* **23** (1969) pp 417–421.

[78] A M Vershik, "The Asymptotic Distribution of Factorizations of Natural Numbers into Prime Divisors", *Soviet Math. Dokl.* **34** (1987) pp 57–61.

[79] G Walsh, "Efficiency vs. Security in Public-Key Cryptography", *ACSC* **21**(3) (1999) pp 81–105.

[80] D Weber, "Computing Discrete Logarithms with the Number Field Sieve", *Algorithmic Number Theory - ANTS II, LNCS* **1122** (1996) pp 99–114.

[81] D Weber, "Computing Discrete Logarithms with Quadratic Number Rings", *Advances in Cryptology - EUROCRYPT 98, LNCS* **1403** (1998) pp 171–183.

[82] D Weber and T Denny, "The Solution of McCurley's Discrete Log Challenge", *Advances in Cryptology - CRYPTO 98, LNCS* **1462** (1998) pp 458–471.

[83] D Weber, *On the Computation of Discrete Logarithms in Finite Prime Fields*, PhD Thesis, Universitat des Saarlandes, 1997.

[84] H Williams, "A Modification of the RSA Public-Key Encryption Procedure", *IEEE Trans. Information Theory*, **26**(6) (1980) pp 726–729.

[85] J Zayer, *Faktorisieren mit dem Number Field Sieve*, PhD Thesis, Universitat des Saarlandes, 1995.