Computational Mathematics Inspired by RSA

submitted by

Nicholas A. Howgrave-Graham

for the degree of Ph.D

of the

University of Bath

1998

Signature of Author ....

Nicholas A. Howgrave-Graham

Summary

The contents of this thesis have been based on the mathematics encountered in public key cryptography; most notably that of the RSA cryptosystem. The thesis itself is split into five main parts, which are outlined below.

Part I: Chapter 1 gives an introduction to cryptography, and also introduces each of the subsequent chapters in detail. The notation used throughout the thesis may be found in Appendix A.

Part II: The second part of the thesis is just one chapter, but it is of a different flavour to the rest of the work. It considers the study of smooth numbers in relation to detecting torsion in a group, and shows that certain smoothness constraints make this problem easier than may be thought.

Part III: The third section of the thesis holds the majority of the work. It starts in Chapter 3 by giving a solid introduction to the theory of lattices, proving numerous results. These results are then applied to finding small solutions of bivariate Diophantine equations in Chapters 4 and 5. The first to be studied are the univariate modular equations, and a novel approach is given for their solution. Chapter 5 then looks at the factorisation equation $xy = N$ showing that this can also be treated in a similar way. The method is extended to allow for factoring over Gaussian integers, factoring of numbers with repeated factors, and analysing factors which lie in residue classes.

Part IV: In Chapter 6 we consider Wiener-type attacks, i.e. exploiting the use of a small private exponent in RSA cryptography. We give an overview of the problem, and then show that if one has many public exponents all corresponding to small private exponents modulo N, then one can improve on Wiener's original attack considerably.

Part V: Finally, in Chapter 7, we summarise in detail the work of this thesis, showing the original results that have been proved and highlighting the problems that remain open.

Acknowledgements

Most notably I would like to thank my supervisor James Davenport for his expert knowledge in so many areas, and his guidance throughout; it has been most kindly appreciated.

I would also like to thank my CASE sponsors, GCHQ, for their financial support, and interest in my PhD. It was an enjoyable and rewarding experience to share my work with them, and I would particularly like to thank Cliff Cocks and Peter Covey-Crump for their encouragement.

I was fortunate enough to spend a little time at IBM, Yorktown Heights nearing the end of the PhD, and for this I am deeply indebted to Don Coppersmith. In fact Don has had a large impact on this thesis, both through the work he has published, and many private communications, and his influence has been most warmly received.

Other people who have directly had a positive impact on the PhD or my interest in mathematics include Marc Joye, Noel Lloyd, James McKee, S. V. Nagaraj, Richard Pinch, Paul Terrill, Jean-Pierre Seifert, and Igor Shparlinski. Thank you all.

Of course I would have gone completely mad without my friends in the Computing Group at the University of Bath and I would particularly like to mention Natee, Dave, Bill, Jeremy, Nam, Jet and John.

Lastly, but perhaps mostly, I'd like to thank Samantha, who's love and general support over the last few years has been my great source of happiness and strength.

Contents

Part I:
Introduction

Chapter 1

Introduction

In this chapter we give an idea of where this thesis stands in the area of cryptology and computational mathematics. To this end the first section gives a broad account of cryptology whilst the second section deals with the RSA cryptosystem in particular. Section 1.3 then details each of the subsequent chapters and their relevance to today's cryptology.

1.1 An overview of cryptology

Before defining *cryptology* let us turn our attention to *cryptography*. Cryptography, particularly public key cryptography, is a fascinating area of current research, in which the work relies heavily on computational results in pure mathematics (especially number theory), combined with practical computing experience.

The history of cryptography is also fascinating, and a good non-technical account of this is given in (Kahn, 1967). More up to date and technical references include[1] (Menezes *et al.*, 1996), (Schneier, 1996) and (Pinch, 1997) from which most of the following information comes.

Cryptography can be defined to be the study of mathematical techniques to achieve *information security goals*. In today's society these goals are diverse, and range from such things as ensuring that messages are being passed without being tampered with (data integrity), or without being understood by unauthorised parties (privacy) to ensuring that one knows who one is talking too (authentication) and that this person

[1] There are of course many more references than these, see for instance the references held within them.

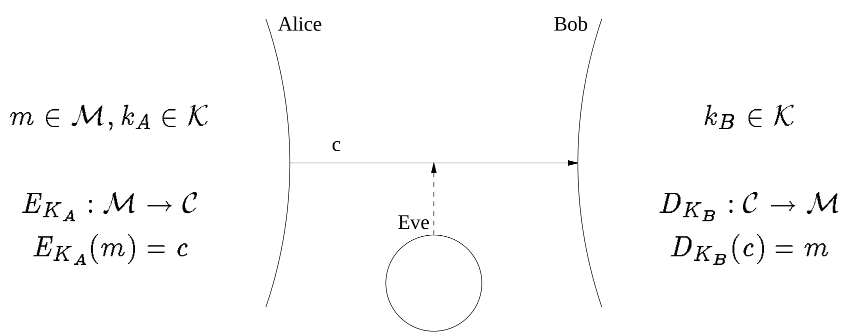
cannot, at a later stage, deny having sent a message they did[2] (non-repudiation).

A particular set of primitives for achieving (some of) these goals is referred to as a *cryptosystem*. A cryptosystem can be measured against criteria such as the goals it meets, the security it offers, its performance and the ease of its implementation.

To meet an information security goal invariably one must describe a *protocol* (i.e. set of rules / distributed algorithm) that the relevant parties should follow. However one must take in to account that there may be malicious parties either "listening in" or editing any information being passed, and also that some of the "legitimate" members of the protocol may also wish to undermine the information security goal.

For instance the following diagram represents the situation where one party (named Alice) wishes to send a message m to a second party (named Bob) ensuring the secrecy of the message, i.e. that any other party (e.g. the one named Eve) cannot recover the message either by *passively* "listening in" to any exchanged information, or *actively* interfering[3] with the exchanged information.

A simple protocol for secrecy (a cipher)



In the above situation it is assumed that Alice and Bob are in possession of keys $K_A, K_B \in \mathcal{K}$ respectively; the set $\mathcal{K}$ is referred to as the *key space*. Let $\mathcal{M}$ denote the set of all possible messages; this is called the *message space*. Alice *encrypts* the message $m \in \mathcal{M}$ by applying an *encryption function* $E_{K_A} : \mathcal{M} \to \mathcal{C}$, $E_{K_A}(m) = c$ dependent on the key K_A. The image, $\mathcal{C}$, of the encryption function is called the *ciphertext space*. Bob then decrypts the message by applying a *decryption function* $D_{K_B} : \mathcal{C} \to \mathcal{M}$ which

[2] i.e. it ought to be possible to prove to a third party that the second party did send the message.

[3] In this simple privacy model an active attack will not help Eve *read* the message m though it might be possible to prevent Bob from receiving a valid message. Also it makes no sense to assume that Bob is malicious since we are sending him the message.

should satisfy

$$D_{K_B}(E_{K_A}(m)) \quad = \quad m \qquad \text{for all } m \in \mathcal{M}, \tag{1.1}$$

for the particular values of K_A and K_B.

It is sound cryptographic practice to assume that the spaces $\mathcal{M}, \mathcal{C}$ and $\mathcal{K}$ are public knowledge[4] and so are the functions E_{K_A} and D_{K_B} for all $K_A, K_B \in \mathcal{K}$. The security of this protocol rests on how hard it is for Eve to deduce $m \in \mathcal{M}$ from $c \in \mathcal{C}$, which is clearly no harder than determining $K_B \in \mathcal{K}$.

In describing protocols throughout this thesis, we will keep to the use of Alice and Bob (A and B) to signify the legitimate parties involved in a protocol, and Eve (the eavesdropper) to denote an attacker (either passive or active).

As mentioned above, the *design* of protocols for achieving information security goals is referred to as *cryptography*. Conversely an analysis of *attacks* on cryptographic protocols is referred to as *cryptanalysis*, and the study of both of these fields is called *cryptology*; as in the title of this section.

The cryptanalysis of ciphers can be aimed either at determining the deciphering key $K_B \in \mathcal{K}$ or uncovering a particular message $m \in \mathcal{M}$ from its ciphertext $c \in \mathcal{C}$. The former of these (key recovery) would imply that Eve was able to read all messages sent to Bob, and achieving this is considered to have completely "broken" the cryptosystem. The latter (specific message recovery) is less disastrous, but is still of serious concern, and one should ensure that neither of these situations can occur with non-negligible probability.

Often in cryptanalysis one assumes that there is further information available to the attacker, for instance one might assume that it is possible to get hold of the (possibly physical) enciphering or deciphering algorithms without knowledge of the underlying keys.

Even if it can be shown that it is infeasible[5] to determine the key K_B or the message m from its ciphertext c the *protocol* may still be attacked. For instance Alice may have been fooled in to the disastrous situation of using a key K_A corresponding to Eve rather than Bob, or perhaps Eve observed Bob's action when a particular message was sent and resends this ciphertext to hopefully induce the same behaviour (without ever

[4] Or rather it is weak to assume this information will remain secret indefinitely.

[5] The infeasibility might be due to assuming the attacker has polynomial time computing power to attack the cryptosystem, or that the problem is at least as hard as a well known hard mathematical problem.

understanding the underlying message). To aid cryptographers prove the security of protocols the use of formal logic on protocol analysis is suggested in (Burrows *et al.*, 1990).

In the cipher above we assumed that Alice and Bob were in possession of keys K_A and K_B such that equation 1.1 was satisfied. However as the attack in the previous paragraph shows, it is imperative that Alice receives the correct key K_A corresponding to Bob's key K_B. Therefore a large and important part of cryptography is to do with issues concerning key management, e.g. the distribution of keys, the updating and storing of keys, and the certification of valid keys.

There are two important branches of cryptography at present; *public-key cryptography* and *private-key cryptography*. In public-key cryptography Bob makes public a key K' which is related to his secret key K_B. For instance in the cipher example above, Alice might use this key as her encryption key K_A (i.e. it satisfies equation 1.1). This sets up a one-way secure line[6] to Bob from anyone interested in sending him a message. It must clearly be infeasible for Eve to deduce K_B from Bob's public key K'. By contrast, in private-key cryptography, one can easily determine the key K_A from K_B and vice versa (in fact they are very often the same key) and thus both must remain secret.

The idea for public key cryptography was first given[7] in 1976 (Diffie & Hellman, 1976), and the first effective public key cryptosystem (rather than simply a protocol for key exchange), RSA, was then discovered in 1978 (see (Rivest *et al.*, 1978)).

At present the challenge facing practical cryptography is to identify the required information security goals and to set in place world standards for secure and efficient protocols which achieve them. In this process it is useful to "build up" protocols from the notion of underlying secure primitives.

1.2 RSA cryptology

As mentioned in Section 1.1 RSA (see (Rivest *et al.*, 1978)) was the first practical public-key cryptosystem invented after the existence of such systems had been speculated in (Diffie & Hellman, 1976). In this section, because of the enormity of the subject, we can only give a summary of the attacks and variations on RSA cryptography. An interested

[6]Often this secure line is used to send a private key, since in practice private key cryptography is often quicker that its public key counterpart.

[7]At least this was the first published result available to the academic community. Readers interested in the non-public history of public-key cryptography (alternatively named non-secret encryption) should visit `http://www.cesg.gov.uk/about/nsecret.htm`

reader is encouraged to look at (Boneh, 1999) and (Joye, 1997) from which most of this information directly comes.

Transmitting secure messages

Let λ denote the Carmichael lambda function, i.e. $\lambda(pq) = \mathrm{lcm}(p-1, q-1)$ for any primes p and q. The RSA protocol is a method for setting up a one-way secure channel to Bob from "anyone else". In order for this to work Bob must, in private, decide upon primes p and q, and a number e, and then (still in private) he calculates $N = pq$ and[8] $d = e^{-1} \pmod{\lambda(N)}$. He then publishes the values of N and e.

Anyone wishing to send a message x to Bob, Alice say, reads Bob's public values of N and e, and then sends $y = x^e \pmod{N}$.

Bob is able to decrypt this message by raising it to the power d modulo N, since $y^d = x^{de} = x \pmod{N}$ by an extension of Fermat's little theorem, i.e. taking d'th powers is equivalent to taking e'th roots modulo $N = pq$ when $de = 1 \pmod{\lambda(N)}$.

Typically Bob would choose the primes p and q to be of approximately the same size, to make the factoring of N as hard as possible. A suitable size for N given todays computing technology might be around 300 decimal digits, or 1024 binary digits. Of course, the reason Bob wishes N to be hard to factor is that if anyone could find p and q then they could calculate d in the same way Bob did, and thus deduce x from the intercepted message $y = x^e \pmod{N}$.

In practice, Bob might wish to decrypt the message y by making use of the Chinese remainder theorem (CRT), i.e. when forming his private information he also calculates $d_p = d \pmod{p-1}$, $d_q = d \pmod{q-1}$ and (from the extended Euclidean algorithm) u and v such that

$$u = \begin{cases} 1 \bmod p \\ 0 \bmod q, \end{cases} \qquad v = \begin{cases} 0 \bmod p \\ 1 \bmod q. \end{cases}$$

Then to find $x = y^d \pmod{N}$ it suffices to find $x_p = y^{d_p} \pmod{p}$ and $x_q = y^{d_q} \pmod{q}$, whereupon $x = ux_p + vx_q \pmod{N}$. This takes approximately a quarter of the time of conventional exponentiation modulo N. However to use this technique notice that Bob must keep the primes p and q, which may be a security risk in itself.

[8] Actually e and d are interchangeable in the sense that Bob could choose d first and then calculate the e that satisfies this property.

Signing messages

Bob's private information may also be used in a protocol for signing messages. If Alice wishes Bob to sign a message x, she could send it to him as plain x, and then he returns $z = x^d \pmod N$. Alice can then verify that Bob signed the message by raising it to the (public) power e modulo N, since $z^e = x^{de} = x \pmod N$.

1.2.1 Other RSA type cryptosystems

The basic idea behind RSA can be extended to other structures, including Lucas sequences (the LUC cryptosystem) and elliptic curves (the KMOV and Demytkov's cryptosystems). It is not the aim of this thesis to discuss these variants, but an interested reader may find details in (Joye, 1997) and (Pinch, 1997).

1.2.2 Factoring large integers

As mentioned above the RSA protocol is no more secure than factoring the integer N. For this reason p and q are chosen to be very large and approximately the same size (but not too close; see for example the attack in Section 5.3), since this "shape" of factorisation seems the hardest to factor.

The most efficient general purpose factoring algorithm presently known is the General Number Field Sieve (GNFS), which has a running time of

$$\exp\Big((c + o(1))m^{1/3}\log^{2/3}(m)\Big)$$

for some $1 < c < 2$ when applied to an m-bit integer. The RSA parameters are chosen so as to make this attack completely infeasible. However if the integer N has a special form, e.g. $(p-1)$ is only comprised of small prime factors, then there may be more efficient factoring methods (see Section 2.3.1). One should make sure that the modulus N is not susceptible to any of the known factoring attacks.

Very recently the 140 digit RSA challenge modulus `RSA-140` was factored using the GNFS in a time estimated to be equivalent to 2000 mips years. It did indeed have two 70 digit prime factors; see (RSA140, 1999) for details.

It is known (and shown for example in (Boneh, 1999)) that knowledge of a pair e and d such that $ed = 1 \pmod{\lambda(N)}$ enables one to factor N. This implies that if two parties are using the same modulus N, supposedly without knowledge of p and q (but with knowledge of their own e_i and d_i), then they are able to factor N and decipher each others messages.

Conversely it is an interesting theoretical question to ask whether being able to take arbitrary e'th roots modulo N leads to a polynomial time factoring of N. It was recently shown in (Boneh & Venkatesan, 1998) that this may not be true in general, in particular for small e, since if it were then one could devise a general polynomial time factoring method (and this is thought unlikely). This means, at least for small e, that breaking RSA may not be as hard as factoring.

1.2.3 Low private exponent attacks

Let us assume that RSA is being used with the standard (i.e. non-CRT) decryption process, and also assume that Bob has chosen a small d to speed up this process. It was shown in (Wiener, 1990) that if $|d| < N^{1/4}$ (and assuming the likely situation that $e/n \approx 1$) then there exists a polynomial time algorithm to find d, and hence to factor N. These kinds of results are discussed in detail in Chapter 6.

1.2.4 Low public exponent attacks

In a similar vein to Section 1.2.3 Bob may wish to ensure that e is relatively small to allow Alice, say, to encrypt information to him quickly.

Coppersmith has shown in (Coppersmith, 1996b) that the small solutions to a univariate modular polynomial may be found in polynomial time using lattice basis reduction, and he further showed in (Coppersmith, 1996a) that similar techniques could factor an integer N given the top $(1/4 + \varepsilon)$ bits of one of its factors. Indeed these results were the main motivation behind the work in Chapters 4 and 5.

One may use these results (and others) on low exponent RSA to discover information in (at least) the following situations.

Partially known plaintext: If Eve can guess the plaintext of an encrypted message apart from one block of data (at a known position in the message) then she can discover this unknown block provided that it is sufficiently small.

Broadcast attack: If Alice wishes to send a message to many people using different RSA moduli, but uses a polynomial of low degree to hide the fact they are the same message, the message may still be uncovered if the number of recipients is large enough.

Related messages: If Alice sends Bob two messages which are related by a known low degree polynomial relationship then both messages may be quickly determined.

Short pad attack: If Alice sends two messages to Bob which only differ by a small amount, say they only differ by a small amount of random padding[9], then the message can still be deciphered.

Partial key exposure attack: If Eve can uncover the bottom $\lfloor n/4 \rfloor$ bits of the decrypting exponent d then she can reconstruct the rest of the bits, and hence factor N.

The details of these attacks are left until Chapters 4 and 6 when we will have developed the necessary tools for their analysis.

1.2.5 Implementation attacks

Any information that Bob decrypts should be treated as secret, even if it does not seem to make any sense at all. This is borne out of the following example due to (Davida, 1982). If Eve intercepts a message $y = x^e \pmod{N}$ to Bob and changes it (referred to as "blinding" the message) to $z = k^e y = (kx)^e \pmod{N}$ for some randomly chosen k, then when Bob calculates $z^d = kx \pmod{N}$ the message (probably) makes no sense. If he discards it in a place accessible by Eve, then all Eve must do is divide by k modulo N to reveal the true message x.

A similar security risk is described in (Bleichenbacher, 1998) it is shown that if an application expects the encrypted information to be a specific format (in this case the (now outdated) PKCS#1 standard), and the application returns an error message if this is not the case, then it is possible to use multiple adaptations of an intercepted message to actually deduce the intercepted message completely. This is a clear warning to keep all information about the decrypted message secret. This includes the situation of signing messages. Bob should only return signed messages that he understands (and agrees to!) since otherwise the technique of blinding can be used to get Bob's signature for something he would not like to apply it to.

Another, more serious, instance of taking care with signing is when the CRT variant of RSA is being employed. In this case, as observed by A. K. Lenstra, if there is a transient fault in the decryption protocol (possibly encouraged by Eve by bombarding

[9] Or perhaps a timestamp.

a smartcard with electromagnetic radiation, or just observed by Alice from a signature that did not verify) then this might allow Eve or Alice to factor N. For example, if a fault occurs during the modulo p exponentiation, but not during the modulo q exponentiation, then the signed message z will be correct modulo p, but not modulo q, i.e.

$$\begin{array}{ll} z = x^d & \pmod p \\ z \neq x^d & \pmod q \end{array} \Rightarrow \begin{array}{ll} z^e = x & \pmod p \\ z^e \neq x & \pmod q \end{array} \Rightarrow \gcd(z^e - x, N) = p.$$

This is very serious since it gives away the factorisation of N, so allows the discoverer to sign fraudulently in the place of Bob. It is however possible and advisable for Bob to take time to check that $z^e = x \pmod N$ before sending the signed message.

A completely different type of attack was devised by Kocher in (Kocher, 1996) using information on the speed of signing known messages to deduce the bits of the decrypting exponent. Rivest has pointed out that this attack can be circumvented by "blinding" the message in a similar way to Davida's attack, so that the attacker no longer knows the messages being decrypted. More recently Kocher also showed that from the power analysis of a smartcard one could deduce the bits of the exponent d since the power consumption for a multi-precision multiplication is above average.

1.3 The structure of the thesis

In this section we detail the structure of the thesis, and the specific interest of each chapter. Note that there is no chapter devoted to background mathematics; rather the background is introduced where necessary. Most notably this occurs at the start of Chapter 2 (smooth numbers, addition chains, and the detection of torsion) and Chapter 3 (lattices). This was done to make the respective chapters more self-sufficient, and not to artificially group together such different areas of mathematics.

The thesis itself is split in to five main parts, of which this introductory chapter is the first; we now describe the remaining parts.

Part II: On detecting group torsion

The second part of the thesis is just one chapter, but it is of a different flavour from the rest of the work. It considers the study of smooth numbers in relation to detecting torsion in a group.

The chapter starts by giving a brief analysis of smooth and semi-smooth numbers, and the Dickman-de Bruijn distribution function.

In the following section a fairly thorough treatment of addition chains for both integers and more importantly sets of integers is given. We start by showing their link with powering algorithms, and show examples of the binary and factor addition chains. We also give reference to the graph theoretical model of addition chains and state the main theorems concerning the function $l(n)$. We then define the concept of T-reliant addition chains to re-place the ideas of (Brickell *et al.*, 1992) in an addition chain framework, and discuss the consequences of this in connection to finding efficient addition chains for sets of integers.

The next section is concerned with the detection of torsion. It starts by defining what we mean by detecting torsion, and then details the use of this in Pollard's $p-1$ factoring algorithm. The large prime variant of this algorithm is discussed, as are the higher order "cyclotomic group" extensions. We also briefly touch on the elliptic curve factorisation method. We then show how the detection of torsion can also aid the breaking of a cryptosystem proposed by Vanstone and Zuccherato, and highlight the slightly different nature of this problem.

The last section of the chapter introduces the notion of deficient numbers, and shows how to classify and produce such numbers. We then show that when artificial smoothness constraints are placed on the size of a group, as in Vanstone and Zuccherato's cryptosystem, the problem of detecting torsion can be speeded up with the use of an algorithm that employs deficient numbers. We also explore the use of parallel processors in this problem.

Part III: Lattice methods for finding small solutions to Diophantine equations

This part of the thesis is by far the largest, and explores the mathematics behind the relatively new lattice methods due to Coppersmith for finding small solutions to various Diophantine equations.

The work starts by explaining lattices in detail, and giving the relevant definitions and theorems used in subsequent chapters. We then consider modular polynomial equations, concentrating mostly on univariate modular equations. Finally we consider the equation $xy = N$, and hence derive some interesting new results on factoring. These chapters are discussed in more detail below.

Lattices

We describe the theory of lattices in some detail. This starts by touching on their roots in linear algebra and explaining the Gram-Schmidt orthogonalisation procedure. We then define the algebraic concept of lattices via the notions of $\mathbb{Z}$-modules and quadratic forms. We show how the use of a concrete basis allows us to move from this (rather platonic) algebraic situation to representing lattice elements as vectors, and quadratic forms and alternative bases as matrices.

The concept of lattice equivalence is then introduced which leads on to a second, equivalent, notion of a lattice, and this is the one maintained throughout the thesis.

The introductory section on lattices concludes by stating some interesting problems concerning lattices and the complexity classes of their solution, and then shows a few of the uses of lattices in present-day computational mathematics and cryptanalysis.

The second section considers the basic properties of lattices, such as the determinant of a lattice, and the successive minima of the quadratic form. The results achieved here are used extensively in later sections.

In the next section we consider the problem of recognising and effectively producing, a *reduced* basis for a lattice. We introduce the notions of a weakly reduced basis, a KZ-reduced basis, and finally an LLL-reduced basis.

We study in detail the implications of a LLL reduced basis, and introduce the slightly weaker notion of an effectively LLL-reduced basis. We then consider the LLL reduction algorithm in some detail and touch on some of its extensions.

We then progress by showing the connection between the LLL reduction criteria and the concept of the dual of a lattice. This has implications on the work in Chapter 4.

The final sections of this chapter then deal with extending the notion of lattice reduction to lattices over number fields. Section 3.5 is independent work, which introduces the notion of unitary lattices (i.e. those over the Gaussian integers), and details all the steps (analogous to those in the integral case) for the working of the LLL algorithm over such structures. However, as explained in Section 3.6, this problem has been previously looked into in the general context of lattices over number fields.

Finding small roots of modular equations

In Chapter 4 we consider the problem of finding small solutions to modular equations; particularly univariate modular equations.

Both Coppersmith's approach and an alternative method are shown to be valid techniques both theoretically, and on a given example. The alternative technique is then explained from a graphical point of view. The connection between the two methods is actually based on the theory of dual lattices and LLL given in Section 3.6, as explained in Section 4.4.

There are a few relatively simple improvements that can be brought to bear on the basic algorithm, and these are discussed in Section 4.5.

The practical results achieved from an implementation of the algorithm written by the author (in `C`) are then given. These show the effect on the running time of the algorithm due to both an increase in the degree of the univariate polynomial, or an increase in the size of the solutions that are being searched for (necessarily less than a theoretical limit). It is also shown that there are optimal choices of the size of solutions searched for, for a given polynomial. Finally the benefits of the small improvements to the algorithm explained in Section 4.5 are analysed.

In the next section of the chapter, we briefly turn our attention to multivariate modular equations, and indeed general Diophantine equations. We explain the approach taken in (Jutla, 1998), and show that a lemma from Chapter 3 slightly improves on this result.

The chapter concludes by showing how the use of finding small solutions to modular polynomial equations affects the security of the RSA cryptosystem. This includes attacks in the following situations:

- when there is only one small block of unknown plaintext,
- (Hastad's) broadcast message situation,
- when a message is repeated with only a small change in random padding,
- when there are many (very) small blocks of unknown plaintext.

The last of these attacks is a new result in this field.

Factoring

In Chapter 5 we analyse the equation $xy = N$ in a similar way to (Coppersmith, 1996a). However we show that one can use a "modular" approach similar to that used in Chapter 4 that has a simpler exposition, and analysis, and removes the need for resultant calculations.

We then extend this factoring algorithm to one over the Gaussian integers, and show how this helps to factor a particular class of integers which are used in a cryptosystem by Vanstone and Zuccherato.

The following section deals with factoring numbers which have repeated divisors. We show that the lattice techniques are particularly effective in this situation, and even for $N = p^2q$ we have an $O(N^{1/8})$ factoring algorithm.

The last section of this chapter consider the problem of finding divisors of an integer N which lie in known residue classes. One method to construct such divisors follows from (Coppersmith, 1996a), but we extend the modular approach to give a simpler method to analyse these divisors and bound the number of them. We compare our results with those previously obtained by H. W. Lenstra.

Part IV: Wiener type attacks on RSA

The fourth part of the thesis considers attacks on RSA when a low decrypting exponent is being used. We describe the general technique of Wiener, and then briefly touch on a better result recently achieved by Boneh and Durfee.

We then consider the approach taken by Guo, in which one studies the related problem of breaking RSA when one has knowledge of more than one encrypting exponent, each with relatively small decrypting exponents modulo a given N. Such a position could possibly occur if one is using different exponents to sign different classes of message, but is content with just one choice of N. We improve on Guo's results and in fact show that as the number of such exponents tends to infinity, one can factor the integer N when the decrypting exponents are as large as $N^{1-\varepsilon}$. However the approach is not feasible with more than about 10 exponents.

Part V: Conclusions and open problems

Finally, in Chapter 7, we summarise in detail the work of this thesis, showing the original results that have been proved and highlighting the problems that remain open.

Appendix A details the mathematical notations and abbreviations that are used throughout the thesis.

Part II:
On Detecting Group Torsion

Chapter 2

Detecting group torsion

In this chapter we aim to give, in Section 2.4, a result that aids the detection of group torsion when we have definite bounds on the smoothness of the group order. Before this result can be described we must give brief overviews of the notions of smooth numbers, addition chains and the detection of torsion; these are the subjects of Sections 2.1, 2.2 and 2.3. The work in Section 2.2 contains the novel notion of T-reliant addition chains.

2.1 Smooth numbers

In this section we define the notion of smooth and semi-smooth numbers, and then discuss the distribution of these numbers. They have been the objects of a considerable amount of study to understand better the probabilities involved in algorithms that use factor bases (see, for example Chapter 9, of (Cohen, 1991)) and for algorithms that detect torsion (see Section 2.3).

The study of smooth numbers began in (de Bruijn, 1951); the criterion for being smooth was that they were "crossed off" during the sieve of Eratosthenes. We may state this more precisely as the following.

Definition 2.1.1 *A number x is said to be y-*smooth *if all the prime divisors of x are $\leq y$.*

For detailed information on the distribution and study of smooth numbers see (Hildebrand, 1986), (Hildebrand & Tenenbaum, 1986), (Moree, 1993) and (Tenenbaum, 1995). However the use of smooth numbers in this chapter does not warrant excessive analysis, and for this reason we only give a brief overview of this subject, based mainly on a short summation in (Pinch, 1997).

Let $S(x,y)$ denote the set of y-smooth numbers up to x (where x is not necessarily an integer), and let $\psi(x,y) = |S(x,y)|$. If we let p be the largest prime $\leq y$, then it is clear that $S(x,y) = S(x,p)$. For any $s \in S(x,p)$ it is either divisible by p or it is not; if it is then s/p is in the set $S(x/p,p)$ otherwise s is in the set $S(x,p')$ where p' is the previous prime to p. This implies that $\psi(x,y) = \psi(x,p') + \psi(x/p,p)$. By repeated use of this idea we have for any z that

$$\psi(x,y) \quad = \quad \psi(x,z) + \sum_{z<p\leq y} \phi(x/p,p), \tag{2.1}$$

(the summation being over prime p only) which is called *Buchstab's* identity.

If we let $u = \log(x)/\log(y)$ and $u \leq 1$ then it is clear that $\psi(x,y) = \lfloor x \rfloor$. The situation is more interesting when $1 < u \leq 2$ in which case (placing $z = x$ in equation 2.1 and hence negating the summation) we obtain

$$\psi(x,y) = \lfloor x \rfloor - \sum_{y<p\leq x} \left\lfloor \frac{x}{p} \right\rfloor \approx x(1 - \log u),$$

the latter approximation coming from the Prime Number Theorem.

By induction on $\lfloor u \rfloor$ it can be shown that $\psi(x,y) \approx x\rho(u)$ where

$$\rho(u) \quad = \quad \rho(k) - \int_k^u \rho(v-1)\frac{\mathrm{d}v}{v}, \tag{2.2}$$

for $k < u \leq k+1$. This is called the *Dickman–de Bruijn* function, and it can be shown that $\rho(u) \approx u^{-u}$ as $u \to \infty$.

In plain terms this means that the fraction of y-smooth numbers less than x depends only on $u = \log(x)/\log(y)$ and is approximately u^{-u}.

The idea of smooth numbers can be extended to the following types of number.

Definition 2.1.2 *A number x is said to be (y,z)-semi-smooth if each of its prime factors is $\leq y$, and all but one are $\leq z$.*

Many algorithms that make use of y-smooth numbers may be adapted slightly to use semi-smooth numbers, and thus be speeded up significantly because of the increased likelihood of x being semi-smooth rather than just smooth (for suitably chosen x, y, z). The adaptations to the algorithms are often referred to as *large prime variants*; see Section 2.3.1 for an example.

One can, as above, define $\psi(x,y,z)$ to be the number of (y,z)-semi-smooth numbers up to x, and to try to work out the distributions for semi-smooth numbers. This has been done in (Bach & Peralta, 1996). The formula for semi-smooth numbers is not quite as simple as that for smooth numbers, but in both cases, due to error terms, it is better to rely on pre-calculated tables. Below we show a few (low precision) results from the table in (Bach & Peralta, 1996) to give an idea of the probabilities involved ($\sigma(v,u)$ is the 2-dimensional equivalent of $\rho(u)$).

$$\psi(t^{10},t) = \psi(t^{10},t,t) \approx \sigma(10,10) \approx 3 \times 10^{-11}$$
$$\psi(t^{10},t^2,t) \approx \sigma(10,5) \approx 5 \times 10^{-9}$$
$$\psi(t^{10},t^3,t) \approx \sigma(10,3.3) \approx 1 \times 10^{-8}$$

2.2 Addition chains

Definition 2.2.1 *An* addition chain *for a positive integer n is a list of integers $a_0,\ldots,a_r$ such that a_0=1, $a_r = n$ and $a_i = a_j + a_k$ for some $j,k < i$. The* length *of the chain is defined to be r, and $l(n)$ denotes the smallest possible length of chain for an integer n.*

Firstly note that one can make an addition chain strictly increasing, i.e. $a_{i+1} > a_i$ by ordering them non-decreasingly and simply removing any duplicate entries. This implies $a_{i+1} < 2a_i$, and therefore

$$l(n) \quad \geq \quad \lceil log_2 n \rceil. \tag{2.3}$$

For example $1,2,4,8,12,14,15$ denotes a chain of length 6 for 15, but actually $l(15) = 5$, as attained by $1,2,3,5,10,15$ or $1,2,3,6,12,15$.

An important application of an addition chain for an integer n is that it implies how to exponentiate "some element" to the n'th power via repeated multiplication. For instance if $x \in \mathbb{Z}_{101}$ say, and one wished to calculate x^{15} then this could be done by 5 multiplications, e.g.

$$\begin{aligned}
y_0 &= x \\
y_1 &= y_0^2 = x^2 \\
y_2 &= y_1 y_0 = x^3 \\
y_3 &= y_2 y_1 = x^5 \\
y_4 &= y_3^2 = x^{10}
\end{aligned}$$

$$y_5 \quad = \quad y_4 y_3 = x^{15}$$

where we can see $y_i = x^{a_i} = x^{a_j + a_k} = x^{a_j} x^{a_k} = y_j y_k$ for some $j, k < i$. An overview of many exponentiation methods is given in Section 14.6 of (Menezes *et al.*, 1996). As we will see below they themselves can conversely be associated with (generalised) addition chains.

The binary addition chain $B(n)$ for an integer n is defined to be

$$B(n) = \begin{cases} 1 & \text{if } n = 1, \\ B(n/2), n & \text{if } n \text{ is even}, \\ B(n-1), n & \text{otherwise}. \end{cases}$$

For instance the longest of the above addition chains for 15 is the binary addition chain for 15. In general the length of a binary addition chain $B(n)$ is $\lfloor \log_2 n \rfloor + \nu(n) - 1$ where $\nu(n)$ is the number of 1's in the binary expansion of n. This implies that the expected length of the binary addition chain is $(3/2)\log_2 n$ and also that $l(n) \leq 2\lfloor \log_2 n \rfloor$. On comparison with equation 2.3 this shows that the binary addition chain is of length at most 2 times longer than the length of the optimal addition chain.

Another common addition chain for an integer n is the factor addition chain $F(n)$, defined as

$$F(n) = \begin{cases} 1 & \text{if } n = 1, \\ F(a), a \times F'(b) & \text{if } n = ab, \\ F(n-1), n & \text{otherwise}, \end{cases}$$

where the list $F'(n)$ is the list $F(n)$ but with the first entry (i.e. 1) omitted, and multiplication on addition chains is defined by

$$k \times \{a_0, \ldots, a_r\} = \{ka_0, \ldots, ka_r\}.$$

Notice that this addition chain implies that $l(ab) \leq l(a) + l(b)$. On average, as quoted in (Knuth, 1981), the length of the factor addition chain is less than that of the binary addition chain.

Other examples of addition chains, a thorough analysis of the function $l(n)$, and a graph theoretic model of addition chains are all given in (Knuth, 1981). A nice summary of the graph theoretical model of addition chains is also described in (Bleichenbacher, 1996) along with techniques for searching for optimal addition chains. Methods for efficiently finding near optimal addition chains are described in (Brlek *et al.*, 1991). It was shown by Brauer in (Brauer, 1939) that $l(n) \leq \log_2(n) + O(\log\log n)$.

The concept of addition chains for a positive integer n may also be extended to a set of positive integers $S = \{n_1, \ldots, n_m\}$.

Definition 2.2.2 *An* addition chain *for a set S of positive integers is an addition chain containing every element of S. Again we denote by $l(S)$ the minimal length of an addition chain for S.*

It was shown (Downey *et al.*, 1981) that finding an optimal addition chain for a set $S = \{n_1, \ldots, n_m\}$ is NP-complete, and the theorem of Brauer was extended in (Yao, 1976) to show that $l(S) \leq \log_2(N) + c\sum_{i=1}^{m} \log_2(n)/(\log_2(\log_2(n_i+2)))$ for some constant c, where $N = \max\{n_i\}_{1 \leq i \leq m}$.

We now introduce slightly more general definitions of addition chains for integers and sets.

Definition 2.2.3 *A T-*reliant *addition chain for a positive integer n is a list of integers $a_1, \ldots, a_r$ such that $a_r = n$ and $a_i = b+c$ for some $b, c \in T \cup A_{i-1}$ where $A_i = \{a_j\}_{1 \leq j \leq i}$ and T is a set of integers. The length of a T-reliant addition chain is defined to be r, and $l_T(n)$ denotes the smallest possible length of chain for the set S with respect to the "stored" set T. A T-reliant addition chain for a set S of positive integers is a T-reliant addition chain such that $S \subseteq T \cup A_r$.*

This definition allows us to "pre-compute" some integers (the ones in the set T), and then form an addition chain for a set, S say, which is allowed to make use of these "stored" values. For example if $T = \{1, 2, 4, 8, 16\}$ and $S = \{2, 7, 10, 25\}$ then the chain $3, 7, 10, 24, 25$ is a T-reliant addition chain for the set S of length 5. Note that an ordinary addition chain for a set S is equivalent to a $\{1\}$-reliant addition chain, with a 1 placed at the start of the chain.

Although we explicitly define T-reliant addition chains for the first time here, they have been implicitly studied in the context of "fast exponentiation with precomputation" in (Brickell *et al.*, 1992). In fact T-reliant addition chains have been defined here to demonstrate the connection of this work with classical addition chains.

Two interesting question regarding T-reliant addition chains are the size of T, and the computational cost of calculating T. The first of these is simply a matter of how much storage one is prepared to use, and the second of these is analogous to the length of an addition chain for the set T.

An interesting observation is that T itself could be formed by being reliant on a pre-pre-computed set T', etc.

In (Brickell *et al.*, 1992) there is no concept of an explicit set S, the problem is rather to form a set T such that one could form a reasonable T-reliant addition chain for any given integer s uniformly distributed on $\{0, \ldots, N-1\}$. Of course if we are explicitly given the set S then the best solution is to find an optimal addition chain for this set S, and there is no concept of T-reliance. However when S is large this seems a hard problem to attack (indeed as mentioned above it is certainly NP-complete), and so the approach taken in (Brickell *et al.*, 1992) seems a practical way to proceed. Put another way, the list $\{t_i\}, \{r_i\}$ may be a reasonably efficient addition chain for S, where $\{t_i\}$ is a reasonable addition chain for T, and $\{r_i\}$ is a reasonable T-reliant addition chain for S.

Clearly one may deem the computational cost of producing T as being significant or not, dependent on memory allowances and the (expected) size of the set S.

In (Brickell *et al.*, 1992) they explain that if

$$n = \sum_{i=0}^{m-1} a_i t_i \tag{2.4}$$

for some set $T = \{t_0, \ldots, t_{m-1}\}$ and where $0 \leq a_i \leq h$, then one may firstly[1] calculate

$$c_j = \sum_{a_i = j} t_i$$

for all $1 \leq j \leq h$, and then calculate

$$\begin{aligned} d_k &= \sum_{j=1}^{k} c_{h+1-j} \\ &= d_{k-1} + c_{h+1-k}. \end{aligned}$$

From which it follows that

$$n = \sum_{k=1}^{h} d_k.$$

For instance if $T = \{1, 5, 10, 22, 30\}$ and $n = 7 \times 1 + 3 \times 5 + 3 \times 10 + 7 \times 22 + 2 \times 30$,

[1] Conceptually this may happen before the following stages of the algorithm, but as shown in (Brickell *et al.*, 1992), one can do all parts together.

and $h = 7$ then

$$\begin{aligned} n &= 7 \times 23 + 3 \times 15 + 2 \times 30 \\ &= 4 \times 23 + 1 \times (23 + 15) + 2 \times (23 + 15 + 30), \end{aligned} \tag{2.5}$$

which we calculate by forming a T-reliant addition chain for $T' = \{15, 23, 30\}$ and then a T'-reliant addition chain for $T'' = \{23, 38, 68\}$, and finally a T''-reliant addition chain for n. Thus the T-reliant addition chain for n is

$$15, 23, 38, 68, 46, 69, 92, 130, 198, 266,$$

of length 10, assuming e.g. 4×23 is simply worked out by adding 23 four times since in general most of the coefficients of equation 2.5 would be 1. It is true that the binary addition chain for 266 has a shorter length, but as we will see for a good choice of set T we may improve markedly on this.

Note that (through T') the length of the T-reliant addition chain for T'' is at most $|T| - 1$ (when, as in this case, all the elements of T are used in the representation of n so we need to add them all to form 68), and the length of the T''-reliant addition chain for n is at most $h - 1$ (when, as in this case, the largest a_i is equal to h). Thus the length of the T-reliant addition chain for n is at most $|T| + h - 2$.

In order to put a number n in the range $\{0, \ldots, N-1\}$ in the form of equation 2.4 the approach taken in (Brickell *et al.*, 1992) considers storing a set of the form

$$T = \{jb^i \;\mid\; 0 \le i \le \lfloor \log_b N \rfloor,\ j \in J\}.$$

for some set $J \subset \{\pm 1, \ldots, \pm(N-1)\}$. They associate with J a set $D(J, h) = \{jk \mid j \in J, 0 \le k \le h\}$ which is designed to be a *basic digit set* (see (Matula, 1982)) for the base b.

Since $D(J, h)$ is a basic digit we may use the algorithm of Matula to find a representation of n of the form

$$\begin{aligned} n &= \sum_{i=1}^{m} a_i' b^i \quad \text{where } a_i' \in D(J, h), \\ &= \sum_{i=1}^{m} a_i j_i b^i \quad \text{for some } 0 \le a_i \le h \text{ and } j_i \in J, \\ &= \sum_{i=1}^{m} a_i t_i \quad \text{for some } t_i \in T. \end{aligned}$$

The choices of base b, set J and integer h (subject to the condition $D(J,h)$ being a basic digit set for the base b) affects the size of T, computational cost of T and computational cost of finding a T-reliant addition chain for a general $n \in \{0,\ldots,N-1\}$. Good choices for b for $N = 2^{160}$ and $N = 2^{512}$ are given in (Brickell *et al.*, 1992) with $[J,h]$ varying from $[\{1\}, b-1]$ to $[\{\pm 1\}, \lfloor (b-1)/2 \rfloor]$ right through to $[\{j \mid 1 \leq j \leq b-1, j \neq k2^{2i+1}\}, 2]$ and $[\{j \mid 1 \leq j \leq b-1\}, 1]$ (the optimal base b increases as the size of J increases).

To give an idea of the savings achieved even when using $[J,h] = [\{1\}, b-1]$ we see that $|T| = \lfloor \log_b N \rfloor + 1$, so we may find a T-reliant addition chain for n of length at most $\lfloor \log_b N \rfloor + b - 2$, and on average of length $\frac{b-1}{b} \lfloor \log_b N \rfloor + b - 2$ assuming $(1/b)$ of the digits of the base b representation for n are zero. For $N = 2^{512}$ the optimal choice of b is 26 in which case the T-reliant addition chain is of length 127.8 on average (132 worst case) rather than the 765 on average (1022 worst case) achieved by the binary addition chain.

In (Brickell *et al.*, 1992) they also explain how this process can be parallelised for a given $n \in \{0,1,\ldots,N-1\}$. From the point of view taken in this section one could therefore define the concept of parallel (T-reliant) addition chains, and express the results in this framework. However we leave this as an exercise for the reader and simply state that with $O(\log N / \log\log N)$ processors the expected time for the necessary additions (i.e. probably the concept of the expected length of parallel addition chains) is $O(\log\log N)$.

Thus far we have ignored the cost of computing the set T. If $T = \{b^i\}_{0 \leq i \leq k}$ where $k = \lfloor \log_b N \rfloor$, then we may firstly find an optimal addition chain for b and then repeatedly use this (akin to the factor addition chain) to produce higher powers of b. The cost of this is small and, as mentioned before, almost definitely better than the binary addition chain for N with expected length $(3/2)\log_2 N$. When J is larger than simply $\{1\}$ we could find a good addition chain for J and use this (again akin to the factor addition chain) to produce jb^i for each $i \in \{0,\ldots,k\}$ and $j \in J$. Of course this second stage could be parallelised by treating each b^i separately. The most costly case when $J = \{1,\ldots,b-1\}$ would entail $(b-1)(k+1)-1$ additions (the cost of producing the b^i being reduced to just one extra addition) which could be reduced to an expected time equivalent to at most $(b-2) + (3/2)\log_2 N$ additions when using $k+1$ processors.

Let us end by summing up the total cost for the two important cases $J = \{1\}$ and $J = \{1,\ldots,b-1\}$:

- With $J = \{1\}$ and for some base b we would expect to form an addition chain for a set S of length at most $(3/2)\log_2 N + v(\frac{b-1}{b}\lfloor \log_b N \rfloor + b - 2)$ when S has v entries uniformly distributed on $\{1,\ldots,N\}$. The number of values that would be needed to be stored is $\lfloor \log_b N \rfloor + 1$.

- With $J = \{1, \ldots, b-1\}$ for some base b we would expect to form an addition chain for a set S of length at most $(b-1)(\lfloor \log_b N \rfloor + 1) - 1 + v(\frac{b}{b-1}\lfloor \log_b N \rfloor)$ when S has v entries uniformly distributed on $\{1, \ldots, N\}$. The number of values that would be needed to be stored is $(b-1)(\lfloor \log_b N \rfloor + 1)$.

For example with $N = 10^{45}$ (which is relevant to the discussion in Section 2.4.1) the optimum choice of b for $J = \{1\}$ is $b = 12$ in which case the addition chain for S will be of length approximately $48.2v + 224.3$. Since $\log_2 N \approx 149.5$ this will be an improvement on finding optimal addition chains for each $s \in S$ whenever $v \geq 3$.

For the case $J = \{1, \ldots, b-1\}$ the optimum value of b is $O(v/\log v)$ i.e. dependent on the size of the set S. This implies the length of the addition chain is $O(v \log_v N)$. If the storage requirement of $O(b \log_b N)$ exceeds a practical upper limit one can simply maximise b with regards to this restriction. For instance with $v = 2.7 \times 10^{14}$ and $b = 10^5$ (again relevant to the discussion in Section 2.4.1) then the addition chain for S will be of length approximately $v \log_b N \approx 9v$ which is over 5 times shorter than using $J = \{1\}$, and almost 17 times shorter than finding optimal addition chains for each $s \in S$.

It is worth remarking that when one has a large set S and many processors it is more efficient to split S amongst the processors rather than making use of the processors in parallel exponentiation methods.

2.3 Detecting group torsion

Definition 2.3.1 *An algorithm $\mathcal{A}$ is said to* detect torsion *in the (multiplicative) group G if given any element $g \in G$ and $n \in \mathbb{Z}$, it can determine if g^n is the identity element of the group.*

This may seem a curious notion, because if one has a group, then it might seem easy to raise it to the power n, using perhaps the addition chain techniques of Section 2.2, and verify if g^n is indeed the identity element. However this assumes that one has a representation of the group in which it is possible to apply the group operation and also to check for the identity of the group.

2.3.1 Pollard's $(p-1)$ factoring method

Pollard was the first to use the detection of torsion to aid the factoring of an integer, $N = pq$ say. He considered the group $(\mathbb{Z}_p^*, \times)$ (where p is prime) and represented the elements (non-uniquely) as elements of $\mathbb{Z}_N^*$. One is able to apply the group operation since multiplication in $\mathbb{Z}_N^*$ respects multiplication in $\mathbb{Z}_p^*$. Further one is able to detect a representation $x \in \mathbb{Z}_N^*$ as the identity element in $\mathbb{Z}_p^*$ since it is either $1 \in \mathbb{Z}_N^*$ or $\gcd(x-1, N) = p$. Therefore, in this group, detection of torsion almost always leads to finding a non-trivial factor of N.

One can pick a random element in $x \in \mathbb{Z}_p^*$ by picking a random element in $\mathbb{Z}_N^*$, but how is one to know what n will be a multiple of the order of x? To answer this Pollard noted that the order of $\mathbb{Z}_p^*$ is $(p-1)$ and he assumed that this was a B-smooth number (i.e. its largest prime factor is less than or equal to B; see Section 2.1). He also assumed a maximum bound w on the size of the highest prime power dividing $(p-1)$. Under these assumptions he suggested using

$$n = \prod_{d \in \mathcal{P}_B} d^{e_d} \tag{2.6}$$

where $\mathcal{P}_B$ is the set of primes $\leq B$, and $d^{e_d} \leq w < d^{e_d+1}$. By the Prime Number Theorem we have $v = |\mathcal{P}_B| \approx B/\log B$.

It is clear that the order of any element $x \in \mathbb{Z}_p^*$ must divide $(p-1)$ which must divide n under the given assumptions. Therefore by picking a random $x \in \mathbb{Z}_N^*$ (as a representation for a random element of $\mathbb{Z}_p^*$) then an algorithm which detects torsion will discover p unless we have the extremely unlikely situation that $x^n = 1 \in \mathbb{Z}_N^*$ (in which case one should choose a different initial x, and repeat the algorithm).

If we assume that $w = B = \sqrt{p}$ (and that $(p-1)$ is as likely to be as smooth as any other number of the same magnitude), then $n \approx w^v \approx \sqrt{p}^{\sqrt{p}/\log\sqrt{p}}$, and therefore using addition chains we will need approximately $\log n \approx \sqrt{p}$ multiplications to detect torsion. The probability of the smoothness assumption being correct is approximately $1 - \log 2 \simeq 0.3$, and as we will see in Section 2.4 the condition on w is likely to be true.

We will now briefly describe an extension to the algorithm so that, without much computational effort, it can detect torsion if the group order is (B', B)-semi-smooth rather than requiring it to be B-smooth; the extension is called the *large prime variant* of Pollard's $(p-1)$ method. One firstly calculates $h = g^n$ as before, and now one is (hopefully) in the position that $h^{m'} = 1$ for *one* prime power $m' \leq B'$. In fact normally $B' < B^2$ in which case m' is actually a prime $\leq B'$ (rather than a prime power). Let

$\Delta(x)$ denote the difference between the prime above x and x itself, then by calculating $g^n \times g^{\Delta(n)} \times g^{\Delta(n+\Delta(n))} \times \ldots$ (in that order) we shall eventually find m'. The saving is made from the fact that the difference between the primes around P are much smaller (i.e. around $\log P$) than P itself, and also that the common values of $g^{\Delta(x)}$ may be stored and reused.

A problem with Pollard's technique is that $(p-1)$ may not be smooth at all, indeed it might be the case that $p-1=2q$ where q is prime. In this case the method, as stated, will not work. However one may use the general technique of detecting torsion on groups other than just Z_p^*.

For instance if one uses the group of elements of $GF(p^2)$ of norm 1; this has order $(p+1)$ and if this is smooth one may find the factor p along similar reasoning to the above. Higher order extensions allow one to generalise this approach to whenever $\Phi_d(p)$ is smooth, where Φ_d is the d'th cyclotomic polynomial. However as the degree of Φ_d increases, so the size of $\Phi_d(p)$ increases, and hence it becomes less likely that $\Phi_d(p)$ is smooth.

Far more usefully, H. W. Lenstra suggested in (Lenstra, 1987) using the elliptic curve groups $E_{a,b}(\mathbb{Z}_p)$, which have order $p+1-t$, where $|t| \leq 2\sqrt{p}$ depends on $a, b \in \mathbb{Z}_p$. Thus by trying many $a, b \in \mathbb{Z}_p$ it is reasonable to assume that a typically smooth (if not better) group order will be attained. The distribution of the order of $E_{a,b}(\mathbb{Z}_p)$ for $a, b \in \mathbb{Z}_p$ is dependent on the class number of $t^2 - 4p$ as explained for example in (Silverman, 1986), and empirical information about this distribution for practical parameter values has been given in (McKee, 1990).

2.3.2 On a cryptosystem of Vanstone and Zuccherato

In (Vanstone & Zuccherato, 1997) they proposed a cryptosystem in which an elliptic curve $E_{a,b}(\mathbb{Z}_N)$ is chosen such that $N = pq$ for two primes p and q, where the number of points on both $E_{a,b}(\mathbb{Z}_p)$ and $E_{a,b}(\mathbb{Z}_q)$ are both B-smooth. It was suggested that $B = 10^{16}$ when p and q were both approximately 10^{75}. Without going in to the details of this cryptosystem we will show in the subsequent sections that having a *known* ceiling[2] on the smoothness of group elements allows one to employ a technique (akin to the large prime variant) that speeds up the detection of torsion.

[2]This is opposed to the situation in Pollard's $(p-1)$ factoring method say, in which we simply make plausible assumptions as to the smoothness of group elements.

2.4 Deficient numbers

In this section we introduce the notion of deficient numbers, and show how they can help the detection of torsion when one has a known bound on the smoothness of a group element, as in Section 2.3.2. In the following let w be an upper bound on the size of the order of the group element; the situation when such a bound is not known in advance is examined in Section 2.4.3.

Definition 2.4.1 *Let $\psi_p(k)$ denote the highest exponent of the prime p such that $p^{\psi_p(k)}$ divides k. A natural number m is called* (w, B)-n-deficient *if for any B-smooth number $s \leq w$ there are at most n primes $p_i \leq B$ for which $\psi_{p_i}(m) < \psi_{p_i}(s)$. When w and B are implicitly defined we will refer to such numbers simply as n-deficient. The number m is called* minimally *(w, B)-n-deficient if this is the least such n with this property (i.e. for some B-smooth number $s \leq w$ there are exactly n primes $p_i \leq B$ for which $\psi_{p_i}(m) < \psi_{p_i}(s)$).*

For instance the number $m = 2^2 \times 3 \times 5 \times 7$ is $(500, 7)$-2-deficient (as shown in Example 2.1 below) and the number in equation 2.6 is (w, B)-0-deficient. The deficient numbers are related to detecting torsion (with known bounds) in the following way: if one has a 0-deficient number m then one can simply detect torsion by raising the group element g to the power m, but if m is a 2-deficient number say, then after calculating $h = g^m$ one only knows that $h^{p^a q^b} = 1$ for some primes $p, q \leq B$ and natural numbers a, b.

We are able to classify the (minimally) deficient numbers exactly.

Theorem 2.4.2 *A natural number m is minimally (w, B)-n-deficient if and only if*

1. *$\prod_{i=1}^{n+1} p_i^{\psi_{p_i}(m)+1} > w$ for all possible prime $(n+1)$-tuples $p_1, \ldots, p_{n+1} \leq B$, and*
2. *there exist n primes $q_1, \ldots, q_n \leq B$ such that $\prod_{i=1}^{n} q_i^{\psi_{q_i}(m)+1} \leq w$.*

The first part classifies exactly the (w, B)-n-deficient numbers, whilst the second part ensures minimality.

Proof: Assume m is (w, B)-n-deficient. If there were $n + 1$ primes $p_1, \ldots, p_{n+1} \leq B$ such that $r = \prod_{i=1}^{n+1} p_i^{\psi_{p_i}(m)+1} \leq w$ then m would be at least $(n+1)$-deficient (since r is B-smooth and $\leq w$), which shows that the first condition must be true. If we further assume that m is minimally (w, B)-n-deficient this implies that there is a B-smooth number $s \leq w$ and n primes $q_1, \ldots, q_n \leq B$ such that $\psi_{q_i}(m) < \psi_{q_i}(s)$. Therefore $\prod_{i=1}^{n} q_i^{\psi_{q_i}(m)+1} \leq \prod_{i=1}^{n} q_i^{\psi_{q_i}(s)} \leq w$ shows that the second condition must also be true.

Conversely we now assume the first condition is true. If w were not (w, B)-n-deficient then there would exist a B-smooth number $s \leq w$ and at least $n+1$ primes $p_i \leq B$ such that $\psi_{p_i}(m) \leq \psi_{p_i}(s)$, so $r = \prod_{i=1}^{n+1} p_i^{\psi_{p_i}(m)+1} \leq \prod_{i=1}^{n+1} p_i^{\psi_{p_i}(s)} \leq w$. This contradicts the first condition, so w must be (w, B)-n-deficient. If we further assume the second condition then this implies that $s = \prod_{i=1}^{n} q_i^{\psi_{q_i(m)}+1}$ is a B-smooth number $\leq w$ which has $\psi_{q_i}(m) < \psi_{q_i}(s)$ for all $1 \leq i \leq n$, so m must be minimally (w, B)-n-deficient. $\square$

Example 2.1 *Let $w = 500$ and $B = 7$, then the B-smooth numbers $\leq w$ are given by*

$$\left\{ 2^{e_2} 3^{e_3} 5^{e_5} 7^{e_7} \leq 500 \;\middle|\; \begin{array}{ccc} 0 & \leq & e_2 & \leq 8 \\ 0 & \leq & e_3 & \leq 5 \\ 0 & \leq & e_5 & \leq 3 \\ 0 & \leq & e_7 & \leq 3 \end{array} \right\}.$$

The number $1260 = 2^2 \times 3 \times 5 \times 7$ can be shown to $(500, 7)$-2-deficient, by considering the following $\binom{4}{3}$ products:

$$\left.\begin{array}{rcl} 2^3 \times 3^2 \times 5^2 & = & 1800 \\ 2^3 \times 5^2 \times 7^2 & = & 9800 \\ 3^2 \times 5^2 \times 7^2 & = & 11025 \\ 2^3 \times 3^2 \times 7^2 & = & 1764 \end{array}\right\} \textit{all more than } 500.$$

The size of them above 500 might suggest that 2520 is $(500, 7)$-1-deficient, but this is shown not to be the case by

$$2^3 \times 3^2 = 72 < 500.$$

More efficiently we only need to find the $(n+1)$ smallest values of $p^{\psi_p(m)+1}$ and verify that the product of all of them is more than w whilst the product of the smallest n is $\leq w$.

Thus we are now in a position to identify a (minimally) (w, B)-n-deficient number. Let us turn our attention to how to efficiently produce one.

Theorem 2.4.3 *Let $\phi_p(w)$ denote the largest power of p such that $p^{\phi_p(w)} \leq w$. The number m will be minimally (w, B)-n-deficient whenever the following two conditions hold.*

1. *For all primes $p \in \mathcal{P}_B$ we have*

$$\psi_p(m) \quad \geq \quad \frac{\phi_p(w)+1}{n+1} - 1,$$

2. *For at least n primes $p \in \mathcal{P}_B$ we have*

$$\psi_p(m) \quad \leq \quad \frac{\phi_p(w)}{n} - 1.$$

The first condition ensures that m is (w, B)-n-deficient whilst the second condition ensures minimality (although neither of these conditions are necessary *for m to be minimally (w, B)-n-deficient).*

Proof: We make use of Theorem 2.4.2. For any prime $(n+1)$-tuple $p_1, \ldots, p_{n+1} \leq B$ we have

$$\begin{aligned} \prod_{i=1}^{n+1} p_i^{\psi_{p_i}(m)+1} \quad &\geq \quad \prod_{i=1}^{n+1} p_i^{\frac{\phi_{p_i}(w)+1}{n+1}} \\ &> \quad \prod_{i=1}^{n+1} w^{\frac{1}{n+1}} \;=\; w, \end{aligned}$$

which shows that m is at least (w, B)-n-deficient. The second condition implies there are n primes $q_1, \ldots, q_n \leq B$ such that

$$\begin{aligned} \prod_{i=1}^{n} q_i^{\psi_{q_i}(m)+1} \quad &\leq \quad \prod_{i=1}^{n} p_i^{\frac{\phi_{q_i}(w)}{n}} \\ &\leq \quad \prod_{i=1}^{n} w^{\frac{1}{n}} \;=\; w, \end{aligned}$$

which shows that m is minimally (w, B)-n-deficient. $\square$

This theorem implies that one can create a (w, B)-n-deficient number of size less than the $(n+1)$'th root of the size of the least (w, B)-0-deficient number.

Example 2.2 *Suppose $w = 3000$ and $B = 13$ and we wish to find a (w, B)-2-deficient number. Notice that $\phi_p(w) = 11, 7, 4, 4, 3, 3$ for $p = 2, 3, 5, 7, 11, 13$, so $2^{11} \times 3^7 \times 5^4 \times 7^4 \times 11^3 \times 13^3 = 19654365366155520000$ is the smallest (w, B)-0-deficient number. Using Theorem 2.4.3 we form the number m with $\psi_p(m) = 3, 2, 1, 1, 1, 1$ for $p = 2, 3, 5, 7, 11, 13$, i.e. $m = 2^3 \times 3^2 \times 5 \times 7 \times 11 \times 13 = 360360$.*

This is not the only method to create a (w, B)-n-deficient number, but a fairly natural approach. Notice that in this case there are many smaller (w, B)-2-deficient numbers, e.g. $m/13$ (seen because $2^4 \times 5^2 \times 13 > 3000$). For this reason one might consider defining the concept of a *reduced* (w, B)-n-deficient number in which none of the prime exponents can be reduced while the number still remains being only n-deficient, or perhaps one might devise an algorithm to find the very least (w, B)-n-deficient number. However for the practical purposes of the following sections the types of (w, B)-n-deficient numbers that Theorem 2.4.3 produces are completely adequate.

2.4.1 The use of 1-deficient numbers

Suppose we have known upper bounds on the size and smoothness of a group element g and we wish to detect torsion (as in the cryptosystem mentioned in Section 2.3.2). We now show how the use of 1-deficient numbers can aid this process.

Algorithm 2.4.4 Torsion detection using (w, B)-1-deficient numbers.

1. Given w and B form a (w, B)-1-deficient number m (as from Theorem 2.4.3).
2. Calculate $h = g^m$ using a standard (near-optimal) addition chain.
3. Let $S = \{p^{\phi_p(w) - \psi_p(m)} \mid p \in \mathcal{P}_B\}$, and form h^s for each $s \in S$ by calculating a T-reliant addition chain for S (as shown in Section 2.2).

The correctness of this algorithm follows since, by the definition of 1-deficiency, $h^{(p^a)}$ is the identity element for some prime $p \in \mathcal{P}_B$ and natural number $a \leq \phi_p(w) - \psi_p(m)$ (due to fact that the the element order is assumed to be $\leq w$).

This algorithm approximately halves the time needed to detect torsion using the classical approach with a (w, B)-0-deficient number m_0 say (as in equation 2.6). To see this note that the size of m_0 is approximately w^v where $v \approx B/\log B$ by the Prime Number Theorem, and so the normal addition chain would be of length about $v \log w$. However the (w, B)-1-deficient number m is approximately the square root of this, and hence has an addition chain of approximately half the length. The remaining part is to find an efficient addition chain for the set S which has v entries around the size of $\sqrt{w}$. If one was to calculate each of these by ordinary (near optimal) addition chains this too would take about $(1/2)v \log w$ multiplications, but we may use the techniques of Section 2.2 to speed this up considerably. For instance assuming that this can be done 10 times faster, means that torsion can be detected in about 0.55 of the time of the classical (w, B)-0-deficient approach. The third stage may also make use of parallel processors to move this ratio nearer to 0.5.

Example 2.3 *Let* $w = 10^{75}$ *and* $B = 10^{16}$ *as is suggested in the cryptosystem in Section 2.3.2, and suppose we want to detect the torsion of a group element* g*. The smallest* (w, B)*-0-deficient number is*

$$m_0 \quad = \quad 2^{249} \times 3^{157} \times \ldots \times 9999999999999937^4,$$

so we form the (w, B)*-1-deficient number*

$$m_1 \quad = \quad 2^{124} \times 3^{78} \times \ldots \times 9999999999999937^2.$$

We then form $h = g^{m_1}$ *using a standard addition chain, which should take approximately half the time of calculating* g^{m_0}*.*

If torsion still has to be detected then it remains to check

$$h^{(2^{125})}, h^{(3^{79})}, \ldots, h^{(9999999999999937^2)}.$$

Let S *be the set of these exponents, i.e.*

$$\begin{array}{lc} \{ & 42535295865117307932921825928971026432, \\ & 49269609804781974438694403402127765867, \\ & \vdots \\ & 9999999999999967000000000000036299999999999998669 \\ & \vdots \\ & 9999999999999987400000000000003969 \quad \} \end{array}$$

so $v = |S| \approx 2.7 \times 10^{14}$*, and the largest element of* S *is approximately* 10^{45}*.*

Let us assume we can store 10^6 *group elements, then we choose a base of* $b = 10^5$ *with* $J = \{1, \ldots, b-1\}$ *and store the set of group elements* h^t *where* $t \in T$ *and* $T = \{jb^i \mid j \in J, 0 \leq i \leq 8\}$*. This entails* 9×10^5 *group multiplications (negligible compared to computing the set* S *below), and storing* 9×10^5 *group elements (deliberately made to be near the storage bound).*

By using the set T *as shown in Section 2.2 each element of* s *can now be calculated in 9 multiplications, and thus all the* h^s *for each* $s \in S$ *in* 2.4×10^{15} *multiplications. Although this is extremely large (hence its use in cryptography), it is small compared to calculating* g^{m_1}*. To see this note that to calculate* g^{m_0} *we need about* $v \log_2 w \approx 6.7 \times 10^{16}$

multiplications, so g^{m_1} needs about half of these at 3.3×10^{16}.

We now explain the connection between the 1-deficient algorithm and the large prime variant algorithm given in Section 2.3.1 . Both have two stages; the first stage forms $h = g^{n_1}$ from which one knows that h^{n_2} is the group identity for some n_2 from a fixed set of integers. In the large prime variant this set is the set of primes (and their powers if applicable) between B_1 and B_2, whilst in the 1-deficient algorithm it is the set of (specific powers of the) primes up to B. This also explains why the two techniques cannot be used together; there would be 2 "missing primes" which would effectively make the situation akin to a 2-deficient problem. We study general n-deficient numbers in the next section.

2.4.2 The use of n-deficient numbers

The technique of the previous section can be extended to general n-deficient numbers thus.

Algorithm 2.4.5 Torsion detection using (w, B)-n-deficient numbers.

1. Given w and B form a (w, B)-n-deficient number m (as from Theorem 2.4.3).
2. Calculate $h = g^m$ using a standard (near-optimal) addition chain.
3. Let $S = \{\prod_{i=1}^{n} p_i^{\phi_{p_i}(w)-\psi_{p_i}(m)} \mid \text{for all prime } n\text{-tuples } p_1, \ldots, p_n \leq B\}$, and form h^s for each $s \in S$ by calculating a T-reliant addition chain for S (as shown in Section 2.2).

Again the correctness of this algorithm follows from the definition of n-deficiency. The second stage is now done in $1/(n+1)$'th of the time needed for the classical (w, B)-0-deficient method, but it is the third stage that becomes a problem. This is because the set S has grown exponentially (compared to the (w, B)-1-deficient algorithm) to be of size $\binom{v}{n}$ with entries about $w^{n/(n+1)}$. Even with $n = 2$ we have $|S| = \binom{v}{2} \approx B^2/(2\log^2 B)$ and so the third stage would take time approximately

$$\frac{2B^2 \log w}{3c \log^2 B},$$

where c is the speed up from the techniques of Section 2.2. The increase in the size of S cannot be outweighed by assuming $c \approx 10$ again; this would lead to a third stage of the algorithm that is far more expensive than the entire classical (w, B)-0-deficient approach. However the (w, B)-n-deficient approach does allows for the third stage to be attacked with multiple processors (e.g. by splitting S amongst them), and if the number of processors exceeds $B/\log B$ then the second stage of the algorithm will become dominant and the entire algorithm will take approximately $1/3$ of the time of the classical method.

In general the n-deficient approach only becomes feasible with the use of $O((B/\log B)^{n-1})$ processors, and implies an algorithm which takes $1/(n+1)$'th of the time taken by the classical approach. Compared with the number of processors this speed up is very small, but there may be cases (e.g. under the given assumptions, and when nothing can be gained from trying another group element g) in which this is the only way to proceed. It is not unreasonable to imagine a parallel architecture that could cope with $n = 2$ and $B \approx 10^6$.

2.4.3 An incremental algorithm

In the above analysis we have assumed we have an upper bound w on the size of the order of the group element g. In this section we briefly examine the situation when one does not know such a bound, but instead knows $w = 10^a$ for some a uniformly distributed on $\{2, \ldots, 20\}$ say.

The (w, B)-0-deficient approach can cope with this altered problem with little change. One would simply assume that w is maximal, i.e. $w = 10^{20}$ in our case, and then as usual calculate g^m where $m = \prod_{p \in \mathcal{P}_B} p^{\phi_p(w)}$. The only difference that care should be taken during the creation of g^m to ensure that each prime power is approximately equal (to take advantage of a smaller w). We will refer to this as the modified (w, B)-0-deficient method.

It is less easy to use the extend the (w, B)-1-deficient approach, though still possible if the range on a is small enough. Assume for some w_0 one had calculated the (w_0, B)-0-deficient number m_0 and the group element $h = g^{m_0}$. We could then use the idea of Algorithm 2.4.4 to "look forward" and detect torsion whenever $w \leq w_0^2$. However if it were the case that $w > w_0^2$ then this "check' (and the associated time) would have been wasted; we cannot, in general, reuse this information.

One can use this approach several times during the modified (w, B)-0-deficient approach, as the following table demonstrates. Here we assume $B = 7$ and $10^2 \leq w \leq 10^{20}$ and we "look ahead" (i.e. calculate the set S) whenever w has grown by a power of $3/2$. We also assume that t_1, the time to create the set S, is $1/10$ of the time t_2, the time to calculate g_{m_0}, and further we assume $t_2 = \log_2 m_0$. The time t_3 is the total time using this scheme (i.e. $t_3 = t_1 + t_2 +$ "all the previous (wasted) t_2's").

w_0	m_0	t_1	w_0^2	S	t_2	t_3
10^2	$2^6 \times 3^4 \times 5^2 \times 7^2$	22.6	10^4	$\{2^6, 3^4, 5^2, 7^2\}$	2.3	24.9
10^3	$2^9 \times 3^6 \times 5^4 \times 7^3$	36.2	10^6	$\{2^9, 3^6, 5^4, 7^3\}$	3.6	42.1
$10^{4.5}$	$2^{14} \times 3^9 \times 5^6 \times 7^5$	56.2	10^9	$\{2^{14}, 3^9, 5^6, 7^5\}$	5.6	67.7
$10^{6.75}$	$2^{22} \times 3^{14} \times 5^9 \times 7^7$	84.7	$10^{13.5}$	$\{2^{22}, 3^{14}, 5^9, 7^7\}$	8.5	104.7
$10^{10.125}$	$2^{33} \times 3^{21} \times 5^{14} \times 7^{11}$	129.7	$10^{20.25}$	$\{2^{33}, 3^{21}, 5^{14}, 7^{11}\}$	13.0	162.7

With this scheme it turns out we would detect torsion, on average, in 0.78 of the time of the modified (w, B)-0-deficient approach when $w = 10^a$ and a is uniformly distributed on $\{2, \ldots, 20\}$. No effort has been made to optimise the frequency of calculating the set S, which should clearly be increased the cheaper "looking ahead" is, i.e. the smaller the ratio t_2/t_1.

Finally note that if there is no upper limit on the size of w (or the range of a is just very large) then the (w, B)-1-deficient approach becomes useless; we simply waste too much time looking ahead.

2.5 Conclusions

In Section 2.2 we introduced the concept of reliant addition chains and showed that this is a generalisation of the standard model of addition chains which allows for precomputation. We then showed how this idea can be used to find an addition chain for a set of natural numbers S using results from (Brickell *et al.*, 1992). We also suggested (though omitted the fine details) that the same kind of model might be useful when describing parallel addition chains.

In Section 2.4 we then defined and classified the (w, B)-n-deficient numbers, and showed their use in detecting torsion when one has a known upper bound on the smoothness of a group element. Although this situation is rather unlikely to happen naturally, it may well be artificially ensured, as is the case for the cryptosystem described in (Vanstone & Zuccherato, 1997). In particular we showed that the (w, B)-1-deficient approach given by Algorithm 2.4.4 approximately halves the time needed to attack this cryptosystem.

We also examined the associated (w, B)-n-deficient algorithms in general in Section 2.4.2, and showed that with many processors they may also be used usefully. However the number of processors grows exponentially for a relatively small increase in speed, so these approaches can only used for (very) small n, and should only be considered when there is no better way to proceed (e.g. one cannot gain anything by trying another group element g etc.).

Part III: Lattice methods for finding small solutions to various bivariate Diophantine equations

Chapter 3

Lattices

In this chapter we introduce the concept of lattices, and give a concrete matrix representation for them that we shall maintain throughout the thesis. We then outline some of their more elementary properties.

Much of the work in the subsequent chapters is based on finding a sufficiently small element of a lattice, and this is achieved through the LLL reduction procedure (see (Lenstra *et al.*, 1982)). This is an algorithm to "reduce" an entire basis of a lattice, but certain properties ensure that a relatively short vector is found also. This algorithm is analysed in detail in Section 3.3.

The original work in this chapter starts in Section 3.4 with an interesting property of the LLL reduction algorithm when applied to a basis of a given lattice or its dual. This result has a large impact on the thesis, implying a new way to look at results in (Coppersmith, 1996b), and influencing all the work in Chapters 4 and 5.

In Section 3.5 the LLL algorithm is shown to extend to structures that we name *unitary lattices*. This work was done completely independently, but similar results were shown in (Fieker & Pohst, 1996). As is shown in Section 5.3 this also has implications on a cryptosystem proposed by Vanstone and Zuccherato.

In Section 3.6 we outline the natural progression of extending LLL to unitary lattices and discuss the direction of current research.

A word of warning is that although every effort has been made to make this chapter easy to read, with examples wherever possible, some of the lattice results may be considered a little theoretical without first examining their applications in the subsequent chapters.

3.1 An introduction to lattices

The theory of lattices was first built up from the 2-dimensional (see (Gauss, 1801)) and via the concept of quadratic forms (see (Lagrange, 1773), (Hermite, 1850), (Korkine & Zolotarev, 1873). A thorough treatment of the algebra of general lattices was later given by Cassels in (Cassels, 1971). However it was not until relatively recently that a large amount of interest has been generated in lattices, primarily via computational number-theoretic problems and the suggested use of knapsack-based cryptosystems. In the last few years this interest has been heightened by Coppersmith's novel use of lattices in finding small solutions to bivariate integer equations. For a good introduction to lattices and/or attacks on knapsack-based cryptosystems see (Joux & Stern, 1998), (Joux, 1993), (Cohen, 1991). It is one of the aims of later chapters of this thesis to discuss the ways in which Coppersmith uses lattices.

Before going any further it should be noted that for a good understanding of lattices one must have a good understanding of basic linear algebra[1]. The Gram-Schmidt orthogonalisation procedure plays a central rôle in this chapter, and for this reason it is introduced here, before the concept of lattices. All the definitions and theorems below have been taken (with little or no changes) from (Cohen, 1991) where the necessary proofs may be found.

Theorem 3.1.1 *Given a basis* $\{b_1, \ldots, b_n\}$ *of the Euclidean space* $\mathbb{R}^n$*, one may form an orthogonal basis* $\{b_1^*, \ldots, b_n^*\}$ *where*

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*, \tag{3.1}$$

and $\mu_{i,j} = (b_i \cdot b_j^*)/||b_j||^2$. *Moreover this new basis satisfies*

$$span\{b_1^*, \ldots, b_i^*\} = span\{b_1, \ldots, b_i\}$$

for all $1 \leq i \leq n$.

One should observe that by dividing the orthogonal vectors by their Euclidean length upon their formation one may also produce an *orthonormal* basis of $\mathbb{R}^n$ with the same span over $\mathbb{R}$.

The mathematical definitions of a quadratic form and a lattice are shown below, but

[1] Having said this I would have thought it perfectly reasonable, and rather rewarding, to learn the two hand in hand.

first we state a general theorem about general $\mathbb{Z}$-modules, to have a context in which to place lattices.

Theorem 3.1.2 *Let V be a finitely generated $\mathbb{Z}$-module (i.e. Abelian group)*

1. *If V_{tors} is the torsion subgroup of V, i.e. the set of elements of $v \in V$ such that there exists $m \in \mathbb{Z}\backslash\{0\}$ with $mv = 0$, then V_{tors} is a finite group, and there exists a non-negative integer n and an isomorphism*

$$V \quad \simeq \quad V_{tors} \times \mathbb{Z}^n$$

   (the number n is called the rank *of V).*

2. *If V is a free $\mathbb{Z}$-module (i.e. if $V \simeq \mathbb{Z}^n$, or equivalently by (1) if $V_{tors} = \{0\}$), then any submodule of V is also free of rank less than or equal to that of V.*

3. *If V is a finite $\mathbb{Z}$-module (i.e. by (1) is V is of zero rank), then there exists n and a submodule L of $\mathbb{Z}^n$ (which is free by (2)) such that $V \simeq \mathbb{Z}^n/L$.*

Definition 3.1.3 *Let K be a field of characteristic different from 2, and let V be a K-vector space. We say that a map $q : V \to K$ is a quadratic form if the following two conditions are satisfied:*

1. *For every $\lambda \in K$ and $x \in V$ we have*

$$q(\lambda \cdot x) = \lambda^2 q(x)$$

2. *If we set $b(x, y) = (1/2)(q(x + y) - q(x) - q(y))$ then b is a (symmetric) bilinear form, i.e. $b(x + x', y) = b(x, y) + b(x', y)$ and $b(\lambda \cdot x, y) = \lambda b(x, y)$ for all $\lambda \in K$, x, x' and y in V (the similar conditions on the second variable follow from the fact that $b(y, x) = b(x, y)$).*

Definition 3.1.4 *A lattice is a pair (L, q) where L is a free $\mathbb{Z}$-module of finite rank and q is a positive definite quadratic form on $L \otimes \mathbb{R}$.*

The above theory has treated lattices rather platonically, i.e. as objects that satisfy certain axioms. They may be treated far more concretely by the introduction of a basis for (L, q), which can imply representations for lattice elements, for alternative bases, and for the quadratic form q.

In the following lemma we have put a bias on row vectors, and on rows of matrices, rather than their column counterparts. This is to better fit in with the uses of lattices developed in the subsequent chapters (and for no deeper reason).

Lemma 3.1.5 *Given a basis $(b_i)_{1\leq i\leq n}$ of a lattice (L, q), where b denotes the symmetric bilinear form associated to q, then*

1. *An element $x \in L$ may be represented by an integer (row) vector $X \in V_n(\mathbb{Z})$ where $x = \sum X_i b_i$. The vector X is often referred to as the* coordinate vector *of x with respect to the basis $(b_i)_{1\leq i\leq n}$.*
2. *An alternative basis $(b'_i)_{1\leq i\leq n}$ may be represented by an integer matrix $H \in GL_n(\mathbb{Z})$ whose rows are the coordinate vectors of the b'_i in terms of the b_i. It follows that the determinant of this matrix must be ± 1 (i.e. $H \in GL_n(\mathbb{Z})$) if and only if $(b'_i)_{1\leq i\leq n}$ is indeed a basis for (L, q).*
3. *It is relatively easily checked that the properties of the quadratic form imply*

$$q(x) \quad = \sum_{1\leq i,j\leq n} q_{i,j} x_i x_j,$$

where $q_{i,j} = b(b_i, b_j)$. This means the quadratic form may be represented by the real positive definite symmetric matrix $Q = (q_{i,j})_{1\leq i,j\leq n}$, and that the associated bilinear form satisfies

$$b(x, y) \quad = \quad Y Q X^t,$$

where X and Y are the (integer) coordinate vectors of x and y respectively. Notice that this means $q(x) = b(x, x) = XQX^t$.

Definition 3.1.6 *We say that two lattices (L, q) and (L', q') are* equivalent *if there is a $\mathbb{Z}$-module isomorphism between L and L' sending q to q'.*

Considered as $\mathbb{Z}$-modules L and L' will be isomorphic if and only if a basis of L maps via an invertible integer matrix (i.e. $H \in GL_n(\mathbb{Z})$) to a basis of L'. For this to map q on to q' means that $Q' = HQH^t$ by Lemma 3.1.5(3). The matrix Q thus gives a representation of a lattice that is unique modulo the equivalence relation $\sim$ where $Q \sim Q'$ if and only if $Q' = HQH^t$ for some $H \in GL_n(\mathbb{Z})$.

The matrix Q is not the only way to represent a lattice as explained now. Given a positive definite symmetric matrix Q we may perform the Cholesky decomposition[2] algorithm to find a matrix $B \in GL_n(\mathbb{R})$ such that $BB^t = Q$. In fact the decomposition ensures that B is the unique (lower) triangular matrix that satisfies this property. It now follows that the lattice $L' = \{y = xB \mid x \in \mathbb{Z}^n\}$ with the Euclidean quadratic norm $q'(y) = \sum y_i^2$ is isomorphic to (L, q). Before we show this let us stress what we have done: We have moved from the situation of having elements of the lattice that are represented by integer vectors, and a complicated quadratic form function (effectively holding the "lattice information") to the situation where this information is now held within the representation of the lattice points (real entried vectors) and the quadratic form is simply Euclidean.

With lattices again being written as the pair (L, Q) but now with L, Q being the matrix representations of their algebraic counterparts, we wish to show that

$$(\{y = xB \mid x \in \mathbb{Z}^n\}, I_n) \quad \simeq \quad (\mathbb{Z}^n, BB^t). \tag{3.2}$$

On consideration this follows immediately from the fact that the application of the quadratic forms to the basis elements coincide.

The above theory gives a justification for the following second definition of a lattice.

Definition 3.1.7 *For a given basis $\{b_1, \ldots, b_n\}$ of $\mathbb{R}^m$ which form the rows of a $(n) \times (m)$ matrix B, a lattice L is defined to be the set of points*

$$L \quad = \quad \{y = xB \mid x \in \mathbb{Z}^n\},$$

together with an associated Euclidean quadratic form $\sum_{i=1}^m y_i^2$.

This is the definition of a lattice (with the matrix B being the implied representation) that we shall be using throughout the thesis. When we refer to the size or more accurately *norm* of a lattice point $x \in L$ we shall mean the square root of the Euclidean quadratic form and denote it $||x||$. Note that this definition does not imply that the basis matrix B is triangular, or even that the basis vectors are of dimension equal to the rank of the lattice (though the rank is obviously a lower bound). However one may clearly enforce these situations by finding the Cholesky decomposition of BB^t.

This representation of a lattice is not unique. One many change the basis, i.e. multiply on the left by any $H \in GL_n(\mathbb{Z})$, and also one may right multiply by an orthonormal

[2] Cholesky decomposition of a matrix MM^t is akin to the Gram-Schmidt orthogonalisation procedure on M (see (Cohen, 1991) for more details). The matrix MM^t is often called the *Gram matrix*.

matrix N, i.e. $N^t = N^{-1}$ since this will not affect $Q = BB^t$ (geometrically this is equivalent to twisting the axes). However the absolute value of the determinant of Q remains unchanged by either of these modifications, and thus is a lattice invariant. Its positive square root (the absolute value of the determinant of B if it is square) is referred to as the determinant Δ of the lattice (L, q).

$$\Delta_{(L,q)} = |\det(B)| = \det(Q)^{1/2}.$$

We shall frequently denote a lattice simply by L, when it will be considered a subset of $\mathbb{R}^n$ with the Euclidean quadratic form. There are many interesting problems associated with lattices, for instance those described below. Finding the complexity classes of algorithms to solve these problems is also very interesting, and for a thorough discussion of NP-completeness and related matters, see for example (Aho *et al.*, 1974).

The shortest vector problem: This is the problem of finding a (non-zero) lattice point with least norm, or in a more generalised form the problem of finding a vector that has a norm that is within some multiple of the smallest one. As we will see in Section 3.3 we may find a vector that has a norm within $2^{(n-1)/2}$ of the smallest one in polynomial time, but the problem of deciding whether a given vector has minimal norm is known to be NP-complete (see for instance (Ajtai, 1998a)).

The closest vector problem: Rather than find the smallest point of a lattice (i.e. the closest non-zero vector to zero), it is also interesting to consider the problem of finding a lattice point which is nearest to some other given point in $\mathbb{R}^n$. As above, the problem can be generalised to finding a vector whose distance from the required point in $\mathbb{R}^n$ is less than some multiple of a closest vector (although the zero vector is allowed as a solution to this problem).

The shortest independent vectors problem: The problem here is to find m linearly independent vectors $v_1, \ldots, v_m$ of the lattice whose length (defined as the norm of the largest one of them) is minimal. Again, one can generalise this to being within a multiple of the shortest length possible for m linearly independent vectors.

The shortest basis problem: The problem here is to find a basis $b_1, \ldots, b_n$ for the lattice that is (within some multiple) the shortest possible (with the length of the basis being defined as the norm of the largest one of them). Note that this is not the same as the shortest independent vectors problem with $m = n$ since n linearly independent vectors do not necessarily generate the lattice.

The complexity issues of these lattice problems has become an active area of research recently, heightened since the publication of (Ajtai, 1998b). This is because it was shown that if one could prove that certain instances of them are NP-hard (in the worst case) it would imply that finding a short element of a lattice from a certain class of lattices was also NP-hard, even in the average case. For a good overview of this area see (Blömer & Seifert, 1999).

The theory of lattices is potentially useful whenever linear dependencies occur. The following list gives a rough, though incomplete, idea of the variety of problems in which lattices have been found to be useful:

Factoring univariate integer polynomials: This is the area that the original LLL paper, (Lenstra *et al.*, 1982), was applied to.

Knapsack-based cryptosystems: Much work has been done on lattice attacks on Knapsack-based cryptosystems, see for example (Joux & Stern, 1998).

Search for linear dependencies: In cryptography one way of attacking a cryptosystem is to make use of unexpected linear dependencies, and the LLL algorithm can be used to spot these, again see (Joux & Stern, 1998).

Minimal polynomials: With an approximation to an algebraic number, one can use the LLL algorithm to guess its minimal polynomial; see (Cohen, 1991) and (Joux & Stern, 1998).

Finding small solutions to bivariate Diophantine equations: This is really what the rest of the thesis is based on. In fact the methods can be used heuristically to find solutions to Diophantine equations in more than 2 variables, but the guaranteed proofs of success fail when applied to these situations.

3.2 Basic properties of lattices

As shown in the previous section, the first and simplest property of a lattice is that it has an invariant determinant. In this section we describe some other basic lattice properties.

In (Cassels, 1971) the important notion of the successive minima of $\|\cdot\|^2$ on lattice points was defined, which we state precisely below.

Definition 3.2.1 *The i'th successive minimum λ_i of a lattice L is the smallest real number r such that there are i linearly independent vectors in L with $\|\cdot\|^2$ at most r.*

We start by showing that there is a maximum bound on λ_1, which depends only on the determinant of the lattice.

Lemma 3.2.2 *There exists a positive constant γ_n such that in any lattice (L, q) with determinant Δ there is an element $x \in L$ such that $q(x) \leq \gamma_n \Delta^{2/n}$.*

Proof: We will actually prove this result constructively (under the assumption of the existence of a least element) by showing $\gamma_n \leq (4/3)^{(n-1)/2}$. Let us first assume we have a two dimensional basis B representing the lattice (L, q), and that $\theta = \lambda_1 = \min_{x \in \mathbb{Z}^n}\{\|xB\|^2\}$. The following diagram shows the steps necessary to show $\gamma_2 \leq \sqrt{4/3}$ (an empty box $\square$ denotes an unspecified entry).

$$B = \begin{pmatrix} \square & \square \\ \square & \square \end{pmatrix} \xrightarrow{HBN^t=} \begin{pmatrix} \sqrt{\theta} & 0 \\ \square & \frac{\det B}{\sqrt{\theta}} \end{pmatrix} \xrightarrow{H'HBN^t=} \begin{pmatrix} \sqrt{\theta} & 0 \\ \boxed{< \frac{1}{2}\sqrt{\theta}} & \frac{\det B}{\sqrt{\theta}} \end{pmatrix}$$

The above diagram is representing the following situation: Let $h_1 \in \mathbb{Z}^n$ be such that $\theta = \|h_1 B\|^2$ is minimal, and let H be any matrix such that $H \in GL_n(\mathbb{Z})$ and H has h_1 as its first row, then we have that $B' = HB$ also represents the lattice (L, q). We now find the orthonormal basis of B' (via the Gram-Schmidt orthogonalisation procedure) which we call N, and then HBN^t is yet another representation our lattice, which is triangular and has top-left entry equal to $\sqrt{\theta}$. One may now perform a second change of basis to ensure that the bottom left entry has absolute size at most $(1/2)\sqrt{\theta}$.

From the assumption that θ corresponds to the the lattice element of minimum size, then we must have

$$\begin{aligned} \theta &\leq (1/4)\theta + \frac{\det(B)^2}{\theta}, \quad \text{so} \\ \theta &\leq \sqrt{\frac{4}{3}}\det B, \quad \text{which proves that } \gamma_2 \leq \sqrt{4/3}. \end{aligned}$$

We show how to extend this result to n dimensions after briefly showing the argument for 3 dimensions:

$$\begin{pmatrix} \sqrt{\theta} & 0 \quad 0 \\ \square & \boxed{\begin{matrix} \\ \frac{\det B}{\sqrt{\theta}} \\ \end{matrix}} \\ \square & \end{pmatrix} \longrightarrow \begin{pmatrix} \sqrt{\theta} & 0 \quad 0 \\ \boxed{< \frac{1}{2}\sqrt{\theta}} & \boxed{< \sqrt{\sqrt{4/3}\frac{\det B}{\sqrt{\theta}}}} \\ \square & \square \quad \square \end{pmatrix}$$

The above diagram represents the situation for a 3-dimensional basis B where we again manage to put $\sqrt{\theta}$ in the top left corner, but now use the above 2-dimensional result to show that we may find a 2-dimensional lattice point of size at most $(2 \det B/\sqrt{3\theta})^{1/2}$ from the sub-lattice of determinant $\det B/\sqrt{\theta}$. From the minimality of θ we now have

$$\begin{aligned} \theta &\leq \frac{1}{4}\theta + \sqrt{\frac{4}{3}\frac{\det B}{\sqrt{\theta}}}, \quad \text{i.e.} \\ \theta &\leq \frac{4}{3}\det B^{3/2}, \quad \text{so } \gamma_3 \leq 4/3. \end{aligned}$$

The general pattern is unfolded by induction; if we assume that

$$\theta_n \leq \left(\frac{4}{3}\right)^{(n-1)/2} \Delta^{2/n},$$

then following the above procedure we can form the equation

$$\begin{aligned} \theta_{n+1} &\leq \frac{1}{4}\theta_{n+1} + \left(\frac{4}{3}\right)^{(n-1)/2} \left(\frac{\Delta}{\sqrt{\theta_{n+1}}}\right)^{2/n}, \quad \text{which implies} \\ \theta_{n+1} &\leq \left(\frac{4}{3}\right)^{n/2} \Delta^{2/(n+1)}, \end{aligned}$$

and thus our induction hypothesis is correct and $\gamma_n \leq (4/3)^{(n-1)/2}$. □

The optimum values of γ_n are only known for $n \leq 8$ and are

$$\gamma_1 = 1,\ \gamma_2^2 = \frac{4}{3},\ \gamma_3^3 = 2,\ \gamma_4^4 = 4,\ \gamma_5^5 = 8,\ \gamma_6^6 = \frac{64}{3},\ \gamma_7^7 = 64,\ \gamma_8^8 = 256,$$

whilst a table for the *best known* bounds are given in (Conway & Sloane, 1988) for all $n \leq 24$.

The above analysis has put an upper bound on the size of λ_1; we now switch our attention to calculating a lower bound for λ_1. Towards this aim let $\{b_1, \ldots, b_n\}$ be a basis of (L, q), and let us consider some properties of the Gram-Schmidt orthogonalisation procedure. From equation 3.1 we know that $\|b_i^*\| \leq \|b_i\|$ and that $b_j \cdot b_j^* = \|b_j^*\|^2$. The first of these properties shows that

$$\Delta = \prod_{i=1}^{n} \|b_i^*\| \leq \prod_{i=1}^{n} \|b_i\|, \tag{3.3}$$

whilst the second property may be used to show that for $r_i \neq 0$

$$\left(\sum_{j=1}^{i} r_j b_j^* = \sum_{j=1}^{i} s_j b_j\right) \Rightarrow (r_i = s_i)\,. \tag{3.4}$$

If we consider a lattice point with minimal norm, $v_1 = xB = x'B^*$, i.e. $||v_1||^2 = \lambda_1$, then although the general entries of $x' \in V_n(\mathbb{R})$ are real we know, from equation 3.4, that the last non-zero entry is the same as the last non-zero entry of x, i.e. an integer. This means that for some $1 \leq i \leq n$ we have

$$\lambda_1 \quad \geq \quad ||b_i^*||^2. \tag{3.5}$$

We now extend these results in three different ways, which are stated now as Lemmas. The last of these appears not to be in the current literature.

Lemma 3.2.3 *Let $\{b_1, \ldots, b_n\}$ be a basis for a lattice L, and let $\{b_1^*, \ldots, b_n^*\}$ be the orthogonal vectors achieved from the Gram-Schmidt orthogonalisation procedure, then for any vector $v \in L$ we have*

$$v \;=\; \alpha_1 b_1 + \ldots + \alpha_n b_n \;=\; \beta_1 b_1^* + \ldots + \beta_n b_n^*$$

where

$$\beta_i = \alpha_i + \sum_{j=i+1}^{n} \mu_{j,i}\alpha_j$$

Proof: Simply take the dot product of v with b_i^*, and rearrange. □

This Lemma shows that although the β_i are real numbers; they are not arbitrary, but rather integer linear combinations of the $\mu_{j,i}$ for $i+1 \leq j \leq n$. This fact is used in Section 3.3.2.

Lemma 3.2.4 *For any given lattice L, and any given basis $\{b_1, \ldots, b_n\}$ of L the i^{th} successive minima satisfies*

$$\lambda_i \quad \geq \quad ||b_j^*||^2$$

for some $i \leq j \leq n$.

Proof: Let $||v_k||^2 = \lambda_k$, we know $\lambda_k \geq ||b_j^*||^2$ where b_j^* is the largest non-zero coefficient of v_k written as a linear combination of the orthogonal vectors. If we have that $j < i$ for all $\{v_k\}_{1 \leq k \leq i}$ then this would contradict the linear independence of the $\{v_k\}$. □

This Lemma is useful when putting upper bounds on the $\|b_i^*\|$ in Section 3.3.1.

Lemma 3.2.5 *For any given lattice L, and any given basis $\{b_1, \ldots, b_n\}$ of L the successive minima satisfy*

$$\prod_{j=1}^{k} \lambda_j \quad \geq \quad \prod_{j=1}^{k} \|b_{i_j}^*\|^2$$

for all $1 \leq k \leq n$, and some unique integers $1 \leq i_j \leq n$.

Proof: Let $\{v_j\}_{1 \leq j \leq k}$ be a set of k linearly independent vectors that attain the λ_j, i.e. $\|v_j\|^2 = \lambda_j$ for all $1 \leq j \leq k$. Further let $S = \{i_1 \ldots i_k\}$ be the set of k largest indices i such that the projections of the vectors $\{v_j\}_{1 \leq j \leq k}$ are linearly independent in the space generated by $\{b_i^*\}_{i \in S}$. Such a set must exists otherwise the vectors $\{v_j\}_{1 \leq j \leq k}$ would be linearly dependent. Let

$$v_j \quad = \quad v_j' + \sum_{h=1}^{k} \alpha_{h,i_j} b_{i_j}^*$$

for $j = 1 \ldots k$, which we write in the matrix form

$$\begin{pmatrix} v_1 \\ \vdots \\ v_k \end{pmatrix} \quad = \quad \begin{pmatrix} v_1' \\ \vdots \\ v_k' \end{pmatrix} + \begin{pmatrix} \alpha_{1,i_1} & \ldots & \alpha_{1,i_k} \\ \vdots & & \vdots \\ \alpha_{k,i_1} & \ldots & \alpha_{k,i_k} \end{pmatrix} \begin{pmatrix} b_{i_1}^* \\ \vdots \\ b_{i_k}^* \end{pmatrix}.$$

We now examine the sub-lattice L' of L generated by the vectors $\{v_j\}_{1 \leq j \leq k}$, and transform the basis $\{v_j\}_{1 \leq j \leq k}$ of L' to $\{w_j\}_{1 \leq j \leq k}$ where

$$\begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{pmatrix} \quad = \quad \begin{pmatrix} w_1' \\ w_2' \\ \vdots \\ w_k' \end{pmatrix} + \begin{pmatrix} \beta_{1,i_1} & & & 0 \\ \beta_{2,i_1} & \beta_{2,i_2} & & \\ \vdots & & \ddots & \\ \beta_{k,i_1} & \beta_{k,i_2} & \ldots & \beta_{k,i_k} \end{pmatrix} \begin{pmatrix} b_{i_1}^* \\ b_{i_2}^* \\ \vdots \\ b_{i_k}^* \end{pmatrix}. \tag{3.6}$$

Such a transformation is possible since the α_{j,i_k} are all integers (by equation 3.4) and thus we can perform integer row operations to leave only one non-zero entry in this column, i.e. β_{k,i_k}. Having done this the rightmost entries of the new rows are also all integers, since they too correspond to lattice points, and so we can repeat the process until we have the triangular matrix above.

From equation 3.6 it is clear that w_1 is such that $\|w_1\| \geq \|b^*_{i_1}\|$ since $\beta_{1,i_1} \in \mathbb{Z}\backslash\{0\}$. From the Gram-Schmidt orthogonalisation procedure we also have that

$$w^*_j = w_j - \sum_{h=1}^{j-1} \mu_{j,h} w^*_j = \beta_{j,i_j} b^*_{i_j} + \dots$$

so $\|w^*_j\| \geq \|b^*_{i_j}\|$ which implies the following

$$\prod_{j=1}^{k} \lambda_j = \prod_{j=1}^{k} \|v_j\|^2 \geq \Delta^2_{L'} = \prod_{j=1}^{k} \|w^*_j\|^2 \geq \prod_{j=1}^{k} \|b^*_{i_j}\|^2$$

which completes the proof. □

Corollary 3.2.6 *Let $\sigma_1, \dots, \sigma_n$ be the non-decreasing values of $\{\|b^*_i\|^2\}_{1 \leq i \leq n}$ for a given basis $\{b_1, \dots, b_n\}$ of a lattice L. Then the successive minima λ_i satisfy*

$$\prod_{i=1}^{m} \lambda_i \quad \geq \quad \prod_{i=1}^{m} \sigma_i.$$

This corollary is useful because one may find lower bounds for $\prod_{j=1}^{k} \lambda_j$ from any basis of L. In the next section the concept of a reduced basis is given, but this result holds for any basis e.g. the original basis that describes the lattice. This fact is used in equation 3.19 to find upper bounds for the sizes of the vectors in a reduced basis.

It should be noted that if one is given a lattice in triangular form (e.g. from the Cholesky decomposition) one can read off, from the diagonal entries, values for $\|b^*_i\|$ (for this particular basis).

3.3 Lattice basis reduction

As was explained in Section 3.1 there are many bases of a lattice (L, q) all equivalent via an invertible integer matrix $H \in GL_n(\mathbb{Z})$. It has been a long-standing mathematical problem to say which of these bases should be considered "reduced".

Naïvely one may think that a basis should be considered reduced if and only if it is comprised of vectors that attain the values λ_i for $1 \leq i \leq n$. However, as this example from (Joux, 1993) shows, it is not as simple as this. If one considers the lattice below generated by the rows of the matrix on the left, it can be noticed that $\lambda_i = 2$ for all $1 \leq i \leq 5$, however there is no way to attain five linearly independent vectors that all have size 2, and that span this lattice. The lattice is certainly not equivalent to the one generated by the rows of the matrix on the right for example (they even have differing determinants).

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

The notions of reduction are strongly linked to the Gram-Schmidt orthogonalisation procedure, and so we shall maintain the notation b_i^* for the orthogonal vectors resulting from this procedure, and $\mu_{i,j} = (b_i \cdot b_j^*)/||b_j||^2$. It may be helpful to keep the following diagram in mind.

$$\begin{aligned} b_1 &= b_1^* \\ b_2 &= \mu_{2,1}b_1^* + b_2^* \\ b_3 &= \mu_{3,1}b_1^* + \mu_{3,2}b_2^* + b_3^* \\ b_4 &= \mu_{4,1}b_1^* + \mu_{4,2}b_2^* + \mu_{4,3}b_3^* + b_4^* \\ &\vdots \end{aligned} \tag{3.7}$$

Perhaps the easiest sense of a reduced basis is that $|\mu_{i,j}| \leq 1/2$ for all $1 \leq j < i \leq n$, and a basis that satisfies this property will be referred to as *weakly reduced.*

To produce a weakly reduced basis notice that by changing the basis by $b_i \leftarrow b_i - mb_j$ this changes $\mu_{i,j}$ in the following way

$$\mu_{i,j} \leftarrow \frac{((b_i - mb_j) \cdot b_j*}{||b_j||^2} = \frac{b_i \cdot b_j*}{||b_j||^2} - m = \mu_{i,j} - m.$$

In such a way we can ensure that $|\mu_{i,j}| \leq 1/2$. However by changing b_i we may have changed all the $\mu_{k,j}$ for $k < i$, so they must then be modified in the same way. Effectively this is simply the Gram-Schmidt orthogonalisation procedure, but we are limiting ourselves to integral changes of base.

Another, far stronger, notion of a reduced basis was introduced by Korkine and Zolotarev in (Korkine & Zolotarev, 1873), which is stated below.

Definition 3.3.1 *A basis* $\{b_1, \ldots, b_n\}$ *of* L *is said to be* KZ-reduced *if, were the Gram-Schmidt orthogonalisation procedure applied to it, the following conditions would hold.*

1. $|\mu_{i,j}| \leq 1/2$ *for all* $1 \leq j < i \leq n$ *(i.e. it is a weakly reduced basis), and*

2. b_i^* *is the shortest non-zero vector in the lattice generated by* $\{\pi_i(b_i), \ldots, \pi_i(b_n)\}$ *where* $\pi_i(b_j)$ *is the projection of* b_j *on to the space* $(\mathbb{R}b_1 + \ldots + \mathbb{R}b_{i-1})^{\perp}$.

In terms of the Gram-Schmidt algorithm and looking at the set of equations 3.7 this definition means that $b_1^* = b_1$ is the shortest vector of the lattice generated by $\{b_1, \ldots, b_n\}$, then, blocking off all the $\mu_{i,1}b_1^*$ terms, b_2^* must be the shortest vector in the lattice generated by $\{b_2^*, \mu_{3,2}b_2^* + b_3^*, \mu_{4,2}b_2^* + \mu_{4,3}b_3^* + b_4^*, \ldots\}$, and so on.

One can produce a KZ-reduced basis if one can find a shortest vector of any given lattice. To do this, one would firstly find the shortest vector of the lattice generated by $\{b_1, \ldots, b_n\}$, and rearrange the basis so that this was b_1. Then one would consider the lattice generated by $\{b_2^*, \mu_{3,2}b_2^* + b_3^*, \mu_{4,2}b_2^* + \mu_{4,3}b_3^* + b_4^*, \ldots\}$, find the smallest element, and change the basis so that this corresponded to b_2, etc.

The problem with a KZ-reduced basis is that it is too hard to produce. As mentioned before finding the shortest vector in a lattice is an NP-complete problem, and thus so is the production of an KZ-reduced basis. In the next section we examine a notion of reduction which has proved far more computationally useful.

3.3.1 The LLL algorithm

The landmark paper of (Lenstra *et al.*, 1982) gave a definition of an *LLL-reduced* basis of L, and more importantly an effective way of computing one in polynomial time.

Definition 3.3.2 *A basis* $\{b_1, \ldots, b_n\}$ *of* L *is said to be* LLL-reduced *if, were the Gram-Schmidt orthogonalisation procedure applied to it, the following conditions would hold.*

$$|\mu_{i,j}| \leq 1/2 \qquad \forall\ 1 \leq j < i \leq n, \tag{3.8}$$

$$\|b_i^* + \mu_{i,i-1}b_{i-1}^*\|^2 \geq (3/4)\|b_{i-1}^*\|^2. \tag{3.9}$$

The second condition is known as the *Lovász condition*, whilst the first will be referred to as the *weakly-reduced condition.* As explained in (Lenstra *et al.*, 1982) the 3/4 in the Lovász condition may actually be replaced with any constant c in the range, $1/4 < c < 1$, though for simplicity we will state all results with respect to $c = 3/4$.

In Section 3.2 we showed that there exists a positive constant γ_n such that in any lattice (L, q) with determinant Δ there is an element $x \in L$ such that $q(x) \leq \gamma_n \Delta^{2/n}$, and that $\gamma_n \leq (4/3)^{(n-1)/2}$. In the following analysis we will show an alternative proof of this statement, with the (slightly weaker) bound of $\gamma_n \leq 2^{(n-1)/2}$, but it is shown in Section 3.3.1 that not only does such a lattice element exists, but we can actually find it in polynomial time.

Implications

Before showing how to produce an LLL-reduced basis we first examine the implications of its definition. Let us start by relaxing the LLL weakly reduced condition to

$$|\mu_{i,i-1}| \quad \leq \quad 1/2 \qquad \forall \;\; 2 \leq i \leq n. \tag{3.10}$$

We call a basis that satisfies conditions 3.10 and 3.9 an *effectively* LLL-reduced basis. This is the only coefficient condition that has an impact on the Lovász condition, and it is unchanged by the algorithm (equivalent to Gram-Schmidt) needed to weakly reduced the remainder of the basis. For this reason one can transform an effectively LLL-reduced basis in to a fully LLL-reduced basis very easily, by simply performing the weak reduction algorithm.

The effectively LLL-reduced conditions imply that

$$||b_{i-1}^*||^2 \quad \leq \quad 2||b_i^*||^2, \tag{3.11}$$

or using this fact repeatedly that $||b_i^*||^2 \leq 2^{j-i}||b_j^*||^2$ for all $i < j$. Restricting ourselves to $i = 1$ we have that for all $1 \leq j \leq n$,

$$||b_1||^2 \quad \leq \quad 2^{j-1}||b_j^*||^2, \tag{3.12}$$

so from equation 3.5, and assuming the worst case (and unlikely) situation that b_n^* is the smallest orthogonal vector, we have

$$||b_1||^2 \quad \leq \quad 2^{n-1}\lambda_1. \tag{3.13}$$

Thus we have shown that if ever one could satisfy the (effectively) LLL-reduced conditions, the vector b_1 would not be "much" larger than a shortest vector of the lattice.

Again considering equation 3.12 but now taking the product with j ranging from 1 up to n we find

$$\|b_1\| \quad \leq \quad 2^{(n-1)/4}\Delta^{1/n}, \tag{3.14}$$

which means that, again assuming one could satisfy the (effectively) LLL-reduced conditions, the vector b_1 would be proof that $\gamma_n \leq 2^{(n-1)/2}$. However note that we have yet to show that the LLL-reduced conditions are actually achievable; this is left to the following sub-section.

In a similar way to the formation of equation 3.14, one may use the inequality in equation 3.11 the other way around to show that

$$\|b_n^*\| \quad \geq \quad 2^{-(n-1)/4}\Delta^{1/n}. \tag{3.15}$$

which is relevant to the discussion in Section 4.2.

Now let us consider the implications of the full LLL-reduced conditions (equations 3.8 and 3.9) These, together with equation 3.1, allow us to show

$$\|b_i\|^2 \quad \leq \quad \frac{1+2^{i-1}}{2}\|b_i^*\|^2 \ \leq \ 2^{i-1}\|b_i^*\|^2, \tag{3.16}$$

and so

$$\Delta \quad \leq \quad \textstyle\prod_{i=1}^{n} \|b_i\| \quad \leq \quad 2^{n(n-1)/4}\Delta, \tag{3.17}$$

and then making use of Lemma 3.2.4 for the right hand inequality one can prove

$$2^{1-i}\lambda_i \quad \leq \quad \|b_i^*\|^2 \ \leq \ \|b_i\|^2 \quad \leq \quad 2^{n-1}\lambda_i. \tag{3.18}$$

Equation 3.18 shows that the vector b_i of an LLL-reduced basis is relatively close to attaining λ_i.

In a similar way to the formation of equation 3.14, we may put the following upper bound on the $\|b_i\|$ for $i \geq 1$.

$$\begin{aligned}
\|b_i\| \quad &\leq \quad 2^{n(n-1)/(4(n+1-i))}\left(\frac{\Delta}{2^{(i-1)(i-2)/2}\prod_{j=1}^{i-1}\|b_j^*\|}\right)^{1/(n+1-i)} \\
&\leq \quad 2^{n(n-1)/(4(n+1-i))}\left(\frac{\Delta}{\left(\prod_{j=1}^{i-1}\lambda_j\right)^{1/2}}\right)^{1/(n+1-i)}
\end{aligned}$$

$$\leq \quad 2^{n(n-1)/(4(n+1-i))} \left(\frac{\Delta}{\left(\prod_{j=1}^{i-1} \sigma_j\right)^{1/2}} \right)^{1/(n+1-i)}, \tag{3.19}$$

where $\sigma_1 \leq \sigma_2 \leq \ldots \leq \sigma_{i-1}$ are the smallest $\|\cdot\|^2$ of the orthogonal vectors w.r.t. any basis of the lattice L (using Lemma 3.2.5). Normally one would be given L w.r.t. some basis, and this result would be used with this basis in mind (and if the basis were triangular then the σ_i are just the squares of the smallest diagonal entries). This result is an extension of one found in (Jutla, 1998), and is made use of in Section 4.7.

It should be mentioned that all the bounds achieved above are only the best that can be proven theoretically; in practice far better bounds are often achieved. For instance if one was actually given an LLL-reduced lattice one could work out from the smallest b_i^* how much smaller the smallest vector could be (for instance if b_1^* were the smallest then b_1 would actually attain λ_1).

We now give a brief summation of these reduction criteria.

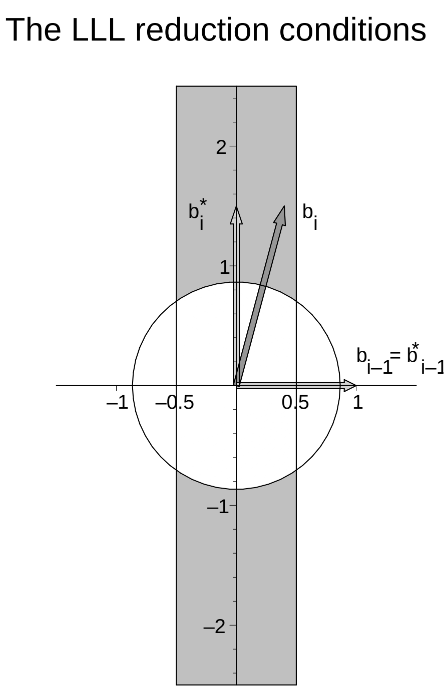


Figure 3.3.3 This diagram represents the real plane which is spanned by the projections of b_{i-1} and b_i on to the space $(\mathbb{R}b_1 + \ldots + \mathbb{R}b_{i-2})^\perp$. It is scaled to the size of 1 : $\mathrm{proj}(b_{i-1})$, but the projections (proj) will be just assumed from now on, i.e. in the diagram and text below. The radius of the circle is $\sqrt{3}/2$. For the vector b_i to satisfy the (full or effective) LLL conditions it must lie in the shaded area (which tends to infinity above and below).

The full LLL conditions ensure that the vectors b_i are fairly orthogonal, i.e. they are fairly close to the b_i^*, and thus the product of their sizes approximates the determinant of the lattice.

The effective LLL conditions simply mean that the b_i^* are not decreasing in size very much, so that $b_1^* = b_1$ is relatively small (and b_n^* is relatively large) w.r.t. the determinant of the lattice.

The reduction algorithm

To show that there is an effective algorithm to find an LLL-reduced basis we will firstly show that for a lattice (L, q) with basis $\{b_1, \ldots, b_n\}$, then the following quantity is lower bounded by a bound dependent only on the lattice itself (not on the particular choice of basis).

$$D \;=\; \left(\|b_1^*\|^2\right)^n \left(\|b_2^*\|^2\right)^{n-1} \cdots \left(\|b_{n-1}^*\|^2\right)^2 \left(\|b_n^*\|^2\right)$$

To see this consider the lattices L_i generated by $\{b_1, b_2, \ldots, b_i\}$ with determinants Δ_i and first minima $\lambda_{1,i}$ for $1 \le i \le n$. Then, by Lemma 3.2.2 for each i we have that

$$\Delta_i^2 \;=\; \prod_{j=1}^{i} \|b_j^*\|^2 \;\ge\; (\lambda_{1,i})^i \, \gamma_i^{-i} \;\ge\; (\lambda_{1,n})^i \, \gamma_i^{-i}$$

so writing $\lambda_{1,n}$ just as λ_1, and noticing $D = \prod_{k=1}^{n} \Delta_i^2$ implies

$$D \;\ge\; (\lambda_1)^{n(n+1)/2} \left(\prod_{i=1}^{n} \gamma_i^i\right)^{-1}$$

which is only dependent on the lattice itself (not any basis of it).

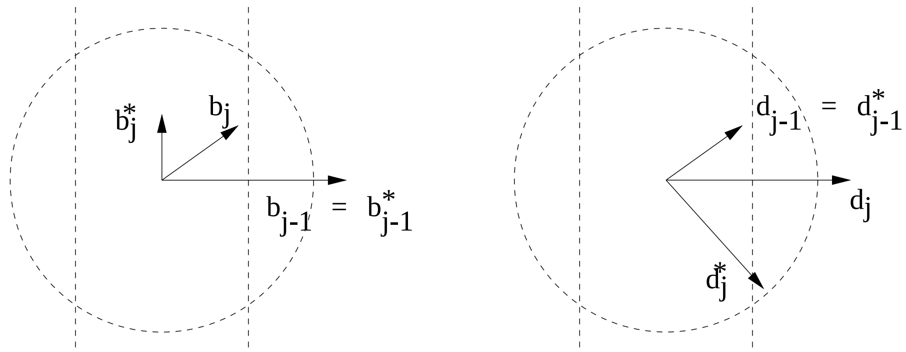
As shown above we can ensure that the basis is weakly reduced very easily, so the only problem in ensuring that the LLL-reduced conditions hold is ensuring that the Lovász condition (equation 3.9) holds for all $2 \le i \le n$. Let us assume that it does not hold between b_{j-1}^* and b_j^*, i.e. b_j^* is somewhere inside the circle and the lines of Figure 3.3.3.

In this situation we would change the lattice basis by swapping b_{j-1} and b_j. If we were to apply the Gram-Schmidt algorithm to this new basis only the vectors b_{j-1}^* and b_j^* would change, since all the other vectors b_i^* would still satisfy the orthogonality and spanning properties of the Gram-Schmidt algorithm. Let us call the new vectors d_{j-1}^* and d_j^*.

Firstly let us note that $\|d_{j-1}^*\|\|d_j^*\|$ must equal $\|b_{j-1}^*\|\|b_j^*\|$ since the determinant of the lattice (the product of the sizes of the orthogonal vectors) is invariant.

Visually we can show this swap by the following diagram, which like Figure 3.3.3 is supposed to represent the projections of the relevant vectors in the space $(\mathbb{R}b_1 + \ldots + \mathbb{R}b_{i-2})^{\perp}$.

Figure 3.3.4 The effect of swapping adjacent vectors

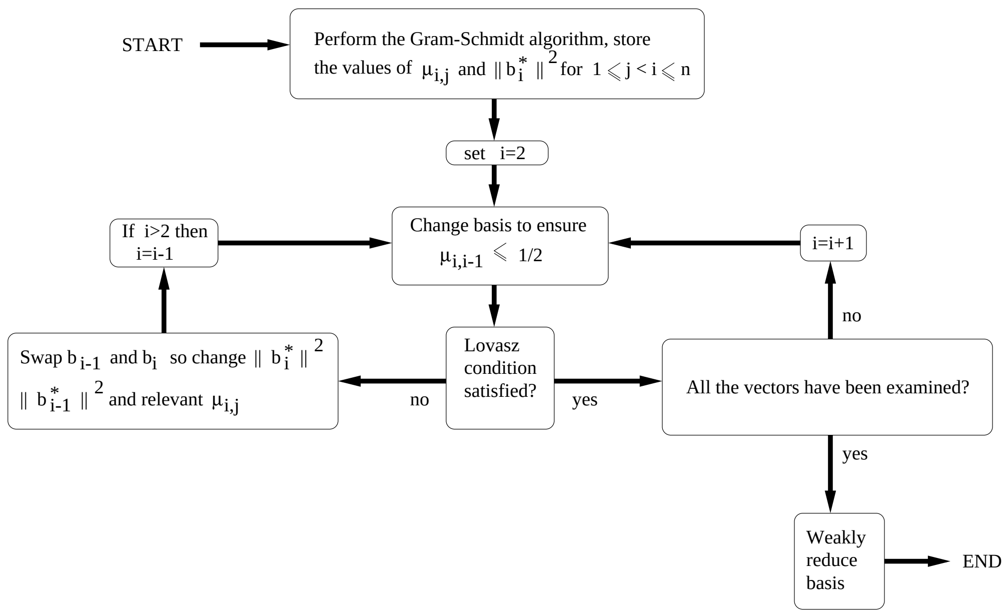


If we let D' denote the new value of D then

$$
\begin{aligned}
D' &= \frac{\left(\|d^*_{j-1}\|^2\right)^{n+2-j}\left(\|b^*_j\|^2\right)^{n+1-j}}{\left(\|b^*_{j-1}\|^2\right)^{n+2-j}\left(\|d^*_j\|^2\right)^{n+1-j}} D \\
&= \frac{\|d^*_{j-1}\|^2}{\|b^*_{j-1}\|^2} D \\
&\leq \frac{3}{4} D
\end{aligned}
$$

We are now in a position to give a polynomial time algorithm to reduce the basis b_i: We could weakly reduce the basis, find the lowest i for which the Lovász condition is not satisfied, swap them, and then repeat this whole process. We could not do this forever since each time we swap, then quantity D falls by $3/4$ and it is lower bounded, thus the algorithm must terminate. Having said this the following flow chart demonstrates a far more efficient algorithm.

Figure 3.3.5 A flow chart of the LLL algorithm



Notice that the Lovász condition is equivalent to

$$||b_i^*||^2 \quad \geq \quad \left(\frac{3}{4} - \mu_{i,i-1}^2\right) ||b_{i-1}^*||^2,$$

so can be computed just from the $||b_i^*||^2$ and $\mu_{i,i-1}$. In fact notice that we only store the $||b_i^*||^2$ and the $\mu_{i,j}$, and not any of the vectors b_i or b_i^*. Of course if one wants to have a matrix representation for the reduced basis one can store and update the vectors b_i throughout the algorithm. Alternatively one might want to have a representation of the reduced basis in terms of the original basis, in which case one can store and update a matrix $H \in GL_n(\mathbb{Z})$ throughout the algorithm.

It can be seen, from looking at Figure 3.3.4, that the necessary changes to $||b_{j-1}^*||^2$, $||b_j^*||^2$, and $\mu_{j,j-1}^2$ when b_{j-1} and b_j are swapped are

$$\begin{aligned}
||d_{j-1}^*||^2 &= ||b_j^*||^2 + \mu_{j,j-1}^2 ||b_{j-1}||^2, \\
||d_j^*||^2 &= \frac{||b_{j-1}^*||^2 ||b_j^*||^2}{||d_{j-1}^*||^2}, \\
\mu'_{j,j-1} &= \frac{||b_{j-1}^*||^2}{||d_{j-1}^*||^2} \mu_{j,j-1},
\end{aligned}$$

where, as above, $||d_{j-1}^*||^2, ||d_j^*||^2, \mu'_{j,j-1}$ denote the new values of $||b_{j-1}^*||^2, ||b_j^*||^2$ and

$\mu_{j,j-1}$ respectively.

The remaining μ's that need to be changed are $\mu_{j-1,k}$, $\mu_{j,k}$ for $1 \le k \le j-2$ and $\mu_{k,j-1}$, $\mu_{k,i}$ for $j+1 \le k \le n$. The $\mu_{j-1,k}$ and $\mu_{j,k}$ simply need to be swapped for $1 \le k \le j-2$. The latter adjustments are slightly more complicated, but again by considering Figure 3.3.4 (picking an arbitrary point for the projection of b_k in to this space, and considering similar triangles) one can show:

$$\begin{aligned} \mu'_{k,j} &= \mu_{k,j-1} - \mu_{j,j-1}\mu_{k,j}, \\ \mu'_{k,j-1} &= \mu_{k,j} + \mu'_{j,j-1}\mu'_{k,j}, \end{aligned}$$

for $j+1 \le k \le n$.

This algorithm will be referred to as the *LLL algorithm* for lattice reduction. It is slightly different from the versions given in (Lenstra *et al.*, 1982) and (Cohen, 1991) because the weak reduction is done in one block at the end, rather than throughout the algorithm, which seems slightly more efficient, although there may be some coefficient explosion with the $\mu_{i,j}$ when $j \ne i-1$ (the effect of this has not be analysed, but for simplicity of exposition the above algorithm is preferred).

When it is applied to an integer basis $b_i \in \mathbb{Z}^n$, $1 \le i \le n$, it can be shown (see (Lenstra *et al.*, 1982)) to have complexity $O(n^6 \log^3 R)$ where n is the dimension of the basis, and $R = \max_{1 \le i \le n}\{\|b_i\|^2\}$. This complexity however, is typically quite pessimistic, and faster times are often achieved in practice. If the entries of the basis are rational, then one can clear denominators before applying the LLL algorithm.

3.3.2 Extensions

When the LLL algorithm is applied to a basis with integral entries, one can ensure, at little expense, that all calculations are done with integers (see for example (Cohen, 1991)). However it is almost always preferable to use LLL with floating point approximations; see (Schnorr, 1988).

A problem with the LLL algorithm, is because the counter i works its way up from 2 to n, the algorithm is sensitive to the order of the initial basis. To combat this problem it was suggested in (Schnorr & Euchner, 1991) that one use "deep insertions", effectively scanning ahead (past b_i) to see which vectors will cancel "nicely" with b_{i-1}.

Although the LLL algorithm frequently finds a shortest element of a lattice (indeed it must if the next shortest linearly independent vector is more than $2^{(n-1)/4}$ times larger) it is not immediately clear how to prove that this is the shortest vector, or find the shortest vector if it is not. An answer to this problem was given in (Kannan, 1983), and is sketched below.

For $v = \alpha_1 b_1 + \ldots \alpha_n b_n = \beta_1 b_1^* + \ldots + \beta_n b_n^*$ to be the smallest vector we must have that it is not larger that b_1, i.e.

$$\beta_n^2 \|b_n^*\|^2 \leq \|v\|^2 \leq \|b_1\|^2 \leq 2^{n-1} \|b_n^*\|^2$$

so $\beta_n \in \mathbb{Z}$ is in the range $-2^{(n-1)/2} \ldots 2^{(n-1)/2}$, and we know that $\alpha_n = \beta_n$.

We can continue this idea and show that $|\beta_{n-1}| \leq 2^{(n-2)/2}$, but β_{n-1} is not typically an integer so it is not immediately obvious that this limits β_{n-1} to a finite search, however this is true from Lemma 3.2.3, which shows that $\beta_{n-1} = \alpha_{n-1} + \mu_{n,n-1}\alpha_n$, and thus α_{n-1} is an integer in the range $-2^{(n-2)/2} - \mu_{n,n-1}\alpha_n \ldots 2^{(n-2)/2} - \mu_{n,n-1}\alpha_n$. Working backwards like this one can find, in the worst case, the smallest vector of a lattice after a linear search of $2^{(n-1)(n-2)/4+1}$ values for the β_i's.

Another interesting extension of the LLL algorithm was given in (Schnorr, 1987) which mixes the ideas of LLL-reduction, and KZ-reduction, and the above idea of Kannan. The resulting (hierarchy of) algorithms are named *blockwise Korkine-Zolotarev* algorithms, because of the introduction of the concept of reduced "blocks" of vectors. Schnorr improves on the efficiency of finding shorter vectors in the lattice.

3.4 The dual lattice and LLL

The dual (or polar) lattice, as given in (Cassels, 1971), is defined as the following.

Definition 3.4.1 *If $\{b_1, \ldots, b_n\}$ is a basis for a lattice L, then there do exist vectors $\{d_1, \ldots, d_n\}$ such that*

$$d_j \cdot b_i = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{3.20}$$

The lattice which is spanned by the basis $\{d_1, \ldots, d_n\}$ is called the dual *lattice of L.*

In terms of matrices, if the rows of B form a basis for a lattice L, then the rows of $(B^{-1})^t$ form a basis (the *dual* basis) for the dual lattice of L. In (Cassels, 1971) the notation L^* and B^* are used for the dual lattice and basis respectively, however to avoid confusion with the Gram-Schmidt procedure we shall adopt the notation L^{-t}

and B^{-t} for these concepts. Notice $B^{-t} = (B^{-1})^t = (B^t)^{-1}$. We now give a theorem linking the dual lattice and the LLL algorithm which is made use of in Section 4.4.

Theorem 3.4.2 *Let the rows of an $(n) \times (n)$ matrix A form a basis for a lattice L, and let B be an effectively LLL reduced basis for this lattice. Further let A^{-t} denote the inverse transpose of A, i.e. $A^{-t} = (A^{-1})^t = (A^t)^{-1}$, and A^r denote the matrix A with the rows reversed. Then the rows of the matrix*

$$D = \left(B^{-t}\right)^r \tag{3.21}$$

form an effectively LLL reduced basis for the dual lattice L^{-t} generated by the rows of A^{-t}.

Moreover, if $\{b_1, \ldots, b_n\}$ and $\{d_1, \ldots, d_n\}$ denote the rows of B and D respectively, then the following relationships hold for all $1 \leq i \leq n$;

$$b_i^* = \frac{d_{n+1-i}^*}{\|d_{n+1-i}^*\|^2}, \tag{3.22}$$

and

$$\frac{b_i \cdot b_{i-1}^*}{\|b_{i-1}^*\|^2} = \frac{d_{n+2-i} \cdot d_{n+1-i}^*}{\|d_{n+1-i}^*\|^2}. \tag{3.23}$$

Proof: To show the rows of D are a basis for L^{-t} at all, let $B = HA$ where $H \in GL_n(Z)$; thus

$$\begin{aligned} D &= \left((HA)^{-t}\right)^r \\ &= \left(H^{-t}A^{-t}\right)^r \\ &= \left(H^{-t}\right)^r A^{-t}, \end{aligned}$$

and $\left(H^{-t}\right)^r \in GL_n(Z)$ as required.

From the definition of the dual lattice we have

$$b_i \cdot d_j = \begin{cases} 1 & \text{if } i + j = n + 1, \\ 0 & \text{otherwise.} \end{cases}$$

By induction on j we have $b_i \cdot d_j^* = 0$ for all $j \leq n - i$, and $b_i \cdot d_{n+1-i} = 1$. Further, since $b_1 = b_1^* = \sum \alpha_{1,i} d_i^*$ with $\alpha_{1,i} = (b_1 \cdot d_i^*)/\|d_i^*\|^2$ this gives $b_1^* = d_n^*/\|d_n^*\|^2$.

Now we assume $b_i^* = d_{n+1-i}^*/\|d_{n+1-i}^*\|^2$ and induct on i. Thus we write $b_{i+1}^* = \sum \alpha_{i+1,j} d_j^*$ where

$$
\begin{aligned}
\|d_j^*\|^2 \alpha_{i+1,j} &= b_{i+1}^* \cdot d_j^* \\
&= \left(b_{i+1} - \sum_{k=1}^{i} \mu_{i+1,k} b_k^* \right) \cdot d_j^* \\
&= b_{i+1} \cdot d_j^* - \sum_{k=1}^{i} \mu_{i+1,k} b_k^* \cdot d_j^*.
\end{aligned}
$$

If $j < n-i$ then both terms on the right hand side are 0, so $\alpha_{i+1,j} = 0$. If $j = n-i$ then $b_{i+1} \cdot d_j^* = 1$ and the terms in the sum are 0, so $\alpha_{i+1,n-i} = 1/\|d_{n-i}\|^2$. Finally if $j > n-i$ then $d_j^* = \|d_j^*\| b_{n+1-j}^*$ by the inductive hypothesis (since $(n+1-j) \leq i$) which implies $\alpha_{i+1,j} = 0$. Thus only $\alpha_{i+1,n-i}$ is non-zero, and so equation 3.22 is true.

With this result we have

$$
\begin{aligned}
\frac{d_{n+2-i} \cdot d_{n+1-i}^*}{\|d_{n+1-i}^*\|^2} &= d_{n+2-i} \cdot b_i^* \\
&= d_{n+2-i}^* \cdot b_i^* \\
&= d_{n+2-i}^* \cdot b_i \\
&= \frac{b_i \cdot b_{i-1}^*}{\|b_{i-1}^*\|^2},
\end{aligned}
$$

which shows equation 3.23 is valid, and hence equation 3.10 holds for the basis D of L', assuming B is itself effectively LLL reduced.

Finally to show equation 3.9 also holds for the basis D of L' when B is effectively LLL reduced, observe that this condition is equivalent to

$$
\|b_i^*\|^2 \geq \left(\frac{3}{4} - \left(\frac{b_i \cdot b_{i-1}^*}{\|b_{i-1}^*\|^2} \right)^2 \right) \|b_{i-1}^*\|^2,
$$

which implies

$$
\begin{aligned}
\frac{1}{\|d_{n+1-i}^*\|^2} &\geq \left(\frac{3}{4} - \left(\frac{d_{n+2-i} \cdot d_{n+1-i}^*}{\|d_{n+1-i}^*\|^2} \right)^2 \right) \frac{1}{\|d_{n+2-i}^*\|^2}, \\
\|d_{n+2-i}^*\|^2 &\geq \left(\frac{3}{4} - \left(\frac{d_{n+2-i} \cdot d_{n+1-i}^*}{\|d_{n+1-i}^*\|^2} \right)^2 \right) \|d_{n+1-i}^*\|^2
\end{aligned}
$$

as required. □

Let the rows of a matrix C form a basis for a lattice L', and suppose that a vector d_n^* is required such that $\|d_n^*\| \geq 2^{-(n-1)/4}\,|\det C|^{1/n}$ for some basis D of L with rows $\{d_1,\ldots,d_n\}$ (i.e. condition 3.15 is required). Clearly the LLL algorithm could find such a d_n^* by reducing the matrix C; the following corollary implies an alternative method.

Corollary 3.4.3 *Let the rows of a matrix C form a basis for a lattice L'. A vector d_n^* such that $\|d_n^*\| \geq 2^{-(n-1)/4}|\det C|^{1/n}$ for some basis D of L' can be found by LLL reducing the matrix C^{-t}.*

Proof: Let $A = C^{-t}$, and apply the LLL to this to form the matrix B. From theorem 3.4.2 we know that $D = \left(B^{-t}\right)^r$ is an effectively LLL reduced basis for C (where $d_n^* = b_1/\|b_1\|^2$), so $\|d_n^*\|^2 \geq 2^{-(n-1)/4}|\det C|^{1/n}$. □

If, as in the method in section 4.2, it is not explicitly the vector d_n^* that is required but a coefficient γ such that $\|vC\| \geq |\gamma|\; 2^{-(n-1)/4}d(L)^{1/n}$, then the following corollary is more useful.

Corollary 3.4.4 *Given a basis C of a lattice L' and a vector $v \in Z^n$, one can find a constant $\gamma \in Z$ such that*

$$\|vC\| \quad \geq \quad |\gamma|\; 2^{-(n-1)/4}d(L')^{1/n}, \tag{3.24}$$

by LLL reducing the matrix C^{-t}.

Proof: As the theory in section 4.2 shows, the normal way to find such a γ is to form an LLL reduced basis D from the initial basis C, and then $\gamma = (v(H')^{-1})_n$ will satisfy equation 3.24, where $D = H'C$.

Instead if we LLL reduce $A = C^{-t}$ to form a basis B, where $B = HA$, and H has rows $\{h_1,\ldots h_n\}$, then

$$\begin{aligned}
\|vC\| &= \|vA^{-t}\| \\
&= \|vH^tB^{-t}\| \\
&= \left\|v\left(H^t\right)^c\left(B^{-t}\right)^r\right\|,
\end{aligned}$$

where $(H^t)^c$ is H^t with its *columns* reversed, and we know $D = (B^{-t})^r$ is an effectively LLL-reduced basis for L'. Thus

$$\begin{aligned}
\|vC\| &\geq \left\|\left(v\left(H^t\right)^c\right)_n d_n^*\right\| \\
&\geq |\gamma|\; 2^{-(n-1)/4}d(L')^{1/n},
\end{aligned}$$

where $\gamma = (v(H^t)^c)_n = v \cdot h_1$. □

This theory suggests that if the LLL algorithm is being used for "something to do with" a large vector d_n^*, it may be better to consider the dual lattice and search for a small vector b_1. The advantage of this is that the LLL algorithm (since it works its way up through the vectors) can have an "early exit" when it has found a small enough b_1, rather than reducing the whole basis to find a large d_n^*.

3.5 Unitary lattices and LLL

In this section we show how to extend the use of the LLL algorithm (see (Lenstra *et al.*, 1982)) from lattices to any free $\mathcal{G}$-module of finite rank (with associated positive definite *unitary* quadratic form), where $\mathcal{G}$ denotes the Gaussian integers. We call such a structure a *unitary lattice*[3], and the extension is analogous to that from an $\mathbb{R}$-inner product space to a $\mathbb{C}$-inner product space in linear algebra.

This work was done completely independently by the author in connection with the work in Section 5.3. It has since come to the authors attention that prior work was done in this area in (Fieker & Pohst, 1996), and consequently in (Schiemann, 1998). This work is discussed in Section 3.6.

In the following theory the symbol $\mathcal{G}$ will be used to denote the Gaussian integers $\{a + ib \mid a, b \in \mathbb{Z}\}$. We now state the definitions and results needed to consider the formation and basis reduction of unitary lattices. The exposition follows very closely that given in Sections 3.1, 3.2 and 3.3.

Definition 3.5.1 *Let V be a $\mathbb{C}$-vector space. A mapping $q : V \to \mathbb{R}$ is called a* unitary quadratic form *if the following two conditions are satisfied:*

1. *For every $\lambda \in \mathbb{C}$ and $x \in V$ we have*

$$q(\lambda x) = |\lambda|^2 q(x).$$

2. *If we set $b(x,y) = \frac{1}{4}(q(x+y) - q(x-y) + iq(x+iy) - iq(x-iy))$ then b is a bilinear form satisfying $b(\lambda x, y) = \lambda b(x,y)$ for all $\lambda \in \mathbb{C}$ and $x, y \in V$.*

Note this definition implies $b(x,x) = q(x) \in \mathbb{R}$ and $b(x, \lambda y) = \overline{\lambda} b(x,y)$. The unitary quadratic form is called *positive definite* if for all non-zero $x \in V$ we have $q(x) > 0$.

[3] We define the concept of a unitary lattice to better fit in with standard linear algebra, however it is acknowledged that unitary lattices can be restated in terms of (higher dimensional) real lattices.

Definition 3.5.2 *A* unitary lattice K *is defined to be a free $\mathcal{G}$-module of finite rank together with a positive definite unitary quadratic form q on $K \otimes \mathbb{C}$.*

With the unitary quadratic form defined as above, we call the bilinear form b a *dot product* and denote it by $x \cdot y = \sum_{i=1}^{n} x_i \overline{y_i}$ for $x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n) \in K$. It is a relatively easy exercise in linear algebra (see, for instance (Stoll & Wong, 1968)) to show that, given a basis $\{b_1, \ldots, b_n\}$ of $\mathbb{C}^n$, the Gram-Schmidt procedure, which is again defined iteratively by

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^* \quad \text{where} \quad \mu_{i,j} = \frac{b_i \cdot b_j^*}{b_j^* \cdot b_j^*},$$

still produces an orthogonal basis $\{b_1^*, \ldots, b_n^*\}$ such that

1. span $\{b_1^*, \ldots, b_i^*\} =$ span $\{b_1, \ldots, b_i\}$ for all $i \leq n$,
2. if Δ_K is the determinant of the unitary lattice K then

$$\Delta_K = \prod_{i=1}^{n} |b_i^*| \, .$$

In a similar way to Lemma 3.1.5 when given a basis of a unitary lattice we can represent certain concepts by vectors and matrices.

Lemma 3.5.3 *Given a basis $(b_i)_{1 \leq i \leq n}$ of a unitary lattice (K, q), where b denotes the symmetric bilinear form associated to q, then*

1. *An element $x \in L$ may be represented by a (row) vector $X \in V_n(\mathcal{G})$ where $x = \sum X_i b_i$. The vector X is often referred to as the* coordinate vector *of x with respect to the basis $(b_i)_{1 \leq i \leq n}$.*
2. *An alternative basis $(b'_i)_{1 \leq i \leq n}$ may be represented by an integer matrix $H \in GL_n(\mathcal{G})$ whose rows are the coordinate vectors of the b'_i in terms of the b_i. It follows that the determinant of this matrix must be ± 1 (i.e. $H \in GL_n(\mathcal{G})$) if and only if $(b'_i)_{1 \leq i \leq n}$ is indeed a basis for (K, q).*
3. *It is relatively easily checked that the properties of the quadratic form imply*

$$q(x) = \sum_{1 \leq i,j \leq n} q_{i,j} x_i x_j,$$

where $q_{i,j} = b(b_i, b_j)$. This means the quadratic form may be represented by the complex positive definite symmetric matrix $Q = (q_{i,j})_{1 \le i,j \le n}$, and that the associated bilinear form satisfies

$$b(x, y) \quad = \quad YQX^t,$$

where X and Y are the (integer) coordinate vectors of x and y respectively. Notice that this means $q(x) = b(x, x) = XQX^t$.

Definition 3.5.4 *We say that two unitary lattices (K, q) and (K', q') are* equivalent *if there is a $\mathcal{G}$-module isomorphism between K and K' sending q to q'.*

Considered as $\mathcal{G}$-modules K and K' will be isomorphic if and only if a basis of K maps via an invertible integer matrix (i.e. $H \in GL_n(\mathcal{G})$) to a basis of K'. For this to map q on to q' means that $Q' = HQH^t$ by Lemma 3.1.5(3). The matrix Q thus gives a representation of a unitary lattice that is unique modulo the equivalence relation $\sim$ where $Q \sim Q'$ if and only if $Q' = HQH^t$ for some $H \in GL_n(\mathcal{G})$.

As in definition 3.1.7 we prefer to define a unitary lattice as a subset of $\mathbb{C}^n$.

Definition 3.5.5 *For a given basis $\{b_1, \ldots, b_n\}$ of $\mathbb{C}^m$ which form the rows of a $(n) \times (m)$ matrix B, a unitary lattice K is defined to be the set of points*

$$K \quad = \quad \{y = xB \ \mid \ x \in \mathcal{G}^m\},$$

together with an associated Euclidean quadratic form $\sum_{i=1}^m y_i \overline{y_i}$.

In a similar way to real lattices we have that the representation of a lattice by the matrix B is only unique up to left multiplication by $H \in GL_n(\mathcal{G})$ and right multiplication by a complex orthonormal matrix N. The invariant determinant of the lattice is thus defined as

$$\Delta_{(K,q)} = |\det(B)| = |\det(Q)|^{1/2}.$$

The concept of the successive minima λ_i of $\|\cdot\|^2$ still holds for unitary lattices. For instance the analogue of Lemma 3.2.4, i.e. $\lambda_i \ge b_j^*$ for some $i \le j \le n$ still holds for unitary lattices.

The notion of an LLL-reduced basis also extends easily to unitary lattices.

Definition 3.5.6 *A basis $\{b_1, \ldots, b_n\}$ of K is said to be* LLL-reduced *if, were the Gram-Schmidt orthogonalisation procedure were applied to it, the following conditions* *would hold.*

$$
\begin{aligned}
|\mu_{i,j}| &\leq 1/\sqrt{2} \qquad \forall\ 1 \leq j < i \leq n, && (3.25)\\
\|b_i^* + \mu_{i,i-1} b_{i-1}^*\|^2 &\geq (3/4)\|b_{i-1}^*\|^2. && (3.26)
\end{aligned}
$$

To understand the change in the weakly reduced condition, notice that if

$$
\mu_{i,j} = \frac{b_i \cdot b_j^*}{\|b_j^*\|^2} = a + ib,
$$

then by performing the change of unitary base $b_i \leftarrow b_i - (\lfloor a \rceil + i \lfloor b \rceil) b_j$ we can transform $\mu_{i,j}$ to

$$
\mu_{i,j}' = \frac{(b_i - (\lfloor a \rceil + i \lfloor b \rceil) b_j) \cdot b_j^*}{\|b_j^*\|^2} = \mu_{i,j} - (\lfloor a \rceil + i \lfloor b \rceil).
$$

This means we can only ensure that $|\mu_{i,j}| \leq \sqrt{(1/2)^2 + (1/2)^2} = 1/\sqrt{2}$.

As before one can relax the weakly reduced condition to just considering $j = i - 1$ and together with the Lovász condition this implies $\|b_{i-1}^*\|^2 \leq 4\|b_i^*\|$. It is then simple to prove

$$
\begin{aligned}
\|b_1\|^2 &\leq 4^{n-1}\lambda_1, \quad \text{and} && (3.27)\\
\|b_1\| &\leq 2^{(n-1)/2}\Delta^{1/n}. && (3.28)
\end{aligned}
$$

The full LLL conditions allow one to show $\|b_i\|^2 \leq 4^{i-1}\|b_i^*\|^2$ and hence

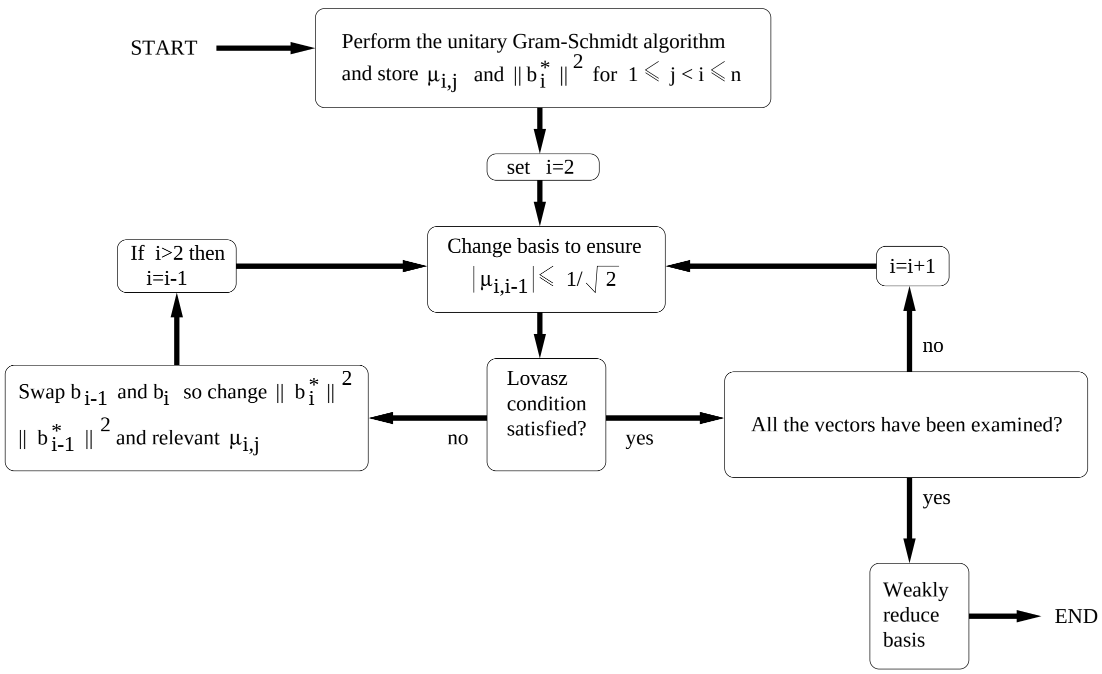
$$
\Delta \;\leq\; \textstyle\prod_{i=1}^n \|b_i^*\| \;\leq\; 2^{(n-1)/2}\Delta, \qquad (3.29)
$$

and

$$
4^{1-i}\lambda_i \;\leq\; \|b_i^*\|^2 \;\leq\; \|b_1\|^2 \;\leq\; 2^{n-1}\lambda_1. \qquad (3.30)
$$

The LLL algorithm follows similarly, and is shown below for completeness sake.

Figure 3.5.7 A flow chart of the LLL algorithm



The complexity of this algorithm is again $O(n^6 \log^3 R)$ where n is the dimension of the basis, and $R = \max_{1 \leq i \leq n}\{\|b_i\|^2\}$, since we have a similar quantity D to the real lattice case, which is lower bounded, and is again multiplied by a factor of at most $3/4$ each time two vectors that fail the Lovász condition are swapped.

3.6 Further extensions of LLL

One can extend the ideas of Section 3.5, and make LLL work over other algebraic structures. For instance it is relatively easy to extend the approach to allow LLL to reduce over $\mathbb{Z}[\alpha] \simeq \{a + \alpha b \mid a, b \in \mathbb{Z}\}$ where α is a degree two algebraic integer, satisfying $\alpha^2 + r\alpha + s = 0$ and $4s - r^2 > 0$. In this case we can define a positive definite norm function $N(a + \alpha b) = a^2 + sb^2 - rab$ and if we also define $\overline{a + \alpha b} = (a - rb) - b\alpha$, then it is left to the reader to verify that the methods of Section 3.5 still hold.

Recently work has been done on extending LLL to work over higher degree algebraic number fields. The started with (Fieker & Pohst, 1996), and more recently has been approached in (Schiemann, 1998). This is an enormous and complicated field of expertise, beyond the scope of this thesis. We refer the interested reader to the above references. However we note that there are complications in determining the complexity of an analogue of the LLL algorithm when working over these higher dimensional algebraic number fields.

Yet another approach to dealing with algebraic integers is to place them within a (standard) integral lattice framework. This has been done in Section 4.7 to help find small solutions to “modular” algebraic integer equation. The benefit of this approach is that the integral LLL algorithm is well understood, and thus the complexity of the method can be well approximated.

Chapter 4

Finding small roots of modular equations

Let $p(x)$ be a univariate modular polynomial of degree k;

$$p(x) \quad = \quad x^k + a_{k-1}x^{k-1} + \ldots + a_1 x + a_0 \pmod{N}. \tag{4.1}$$

It is assumed that $p(x)$ is monic and irreducible, and that N is not prime, but hard to factor[1]. The following theorem was proved in (Coppersmith, 1996b).

Theorem 4.0.1 *If $p(x)$ is a univariate polynomial of degree k, then for any modulus N all the solutions $p(x_0) = 0 \pmod{N}$ with $|x_0| < N^{1/k}$ may be found in time polynomial in $\log N$ and k.*

In this chapter we describe another computational method for the proof of this theorem i.e. how to find all the small integral roots, $x_0 \in \mathbb{Z}$, of equation 4.1, and show the relationship between the approach taken here, and that taken in (Coppersmith, 1996b). It will be proved, via the general result on dual lattices developed in Section 3.4 that these two algorithms are in fact equivalent, though the present approach is preferred for mathematical simplicity (and therefore ease of implementation), and also, very slightly, for efficiency reasons.

It has been shown in (Coppersmith, 1996b), and (Coppersmith *et al.*, 1996), that finding small solutions to equation 4.1 can lead to various attacks on the RSA cryptographic scheme when using a small encrypting exponent. These attacks are outlined

[1]These assumptions are only to prevent there being easier ways to attack the problem. Only the fact that $p(x)$ is monic is important in what follows.

in Section 4.8. However we start in sections 4.1 and 4.2 by giving expositions of the algorithms in question, together with proofs of their validity. Examples of both algorithms are then shown in section 4.3, together with a pictorial explanation of the new method.

In Section 4.4 we show that it is indeed the theory of Section 3.4 that links the two methods. Section 4.5 then gives two slight practical improvements to the basic algorithm.

Implementation considerations are discussed in Section 4.6, with timings results given for a `C` implementation written by the author, using `Gnu MP` for a multi-precision integer package. Reference is also given to more extensive results recently achieved.

In Section 4.7 we extend the basic idea of Section 4.1 to finding small roots of *algebraic* modular equations.

As mentioned above, in Section 4.8 we conclude by examining the applications of this work to cryptography. This section contains a novel result in which we show that there are provably weak places to hide information even when splitting this between many blocks.

4.1 The method

In this section we give an exposition of a new method for finding the small roots of a (monic) univariate modular equation $p(x) = 0 \pmod{N}$. The work has been published in (Howgrave-Graham, 1997).

Observe that for any polynomial $r(x)$, and natural number X, we have the following upper bound on the absolute size of $r(x)$ in the region $|x| \leq X$.

$$\begin{aligned} |r(x)| &\leq |x^k| + |a_{k-1}x^{k-1}| + \ldots + |a_1 x| + |a_0| \\ &\leq |X^k| + |a_{k-1}X^{k-1}| + \ldots + |a_1 X| + |a_0| \quad \text{for all } |x| \leq X. \end{aligned}$$

For some integer $h \geq 2$, and natural number X we define a lower triangular $(hk) \times (hk)$ matrix $M = (m_{i,j})$. The entry $m_{i,j}$ is given by $e_{i,j}X^{j-1}$, where $e_{i,j}$ is the coefficient of x^{j-1} in the expression

$$q_{u,v}(x) = N^{(h-1-v)}x^u(p(x))^v, \tag{4.2}$$

with $v = \lfloor (i-1)/k \rfloor$, and $u = (i-1) - kv$. Notice that $q_{u,v}(x_0) = 0 \pmod{N^{h-1}}$ for all $u, v \geq 0$. All other entries of the matrix are zero, and so this matrix has determinant

$$\det M = X^{hk(hk-1)/2} N^{hk(h-1)/2}. \tag{4.3}$$

Let B be an LLL-reduced basis of the rows of M, and denote the first (small) row vector of B by b_1. Equation 3.14 implies that

$$\|b_1\|_2 \leq 2^{(hk-1)/4} X^{(hk-1)/2} N^{(h-1)/2}. \tag{4.4}$$

Letting $b_1 = cM$ also gives

$$\begin{aligned}
\|b_1\|_2 &\geq \frac{1}{\sqrt{hk}} \|b_1\|_1 \\
&= \frac{1}{\sqrt{hk}} \left(\left| \sum_{i=1}^{hk} c_i m_{i,1} \right| + \left| \sum_{i=1}^{hk} c_i m_{i,2} \right| + \ldots + \left| \sum_{i=1}^{hk} c_{hk} m_{i,hk} \right| \right) \\
&= \frac{1}{\sqrt{hk}} \left(\left| \sum_{i=1}^{hk} c_i e_{i,1} \right| + \left| \left(\sum_{i=1}^{hk} c_i e_{i,2} \right) X \right| + \ldots + \left| \left(\sum_{i=1}^{hk} c_{hk} e_{i,hk} \right) X^{hk-1} \right| \right) \\
&\geq \frac{1}{\sqrt{hk}} \, |r(x)| \quad \text{for all } |x| \leq X,
\end{aligned} \tag{4.5}$$

where

$$\begin{aligned}
r(x) &= \sum_{i=1}^{hk} c_i e_{i,1} + \left(\sum_{i=1}^{hk} c_i e_{i,2} \right) x + \ldots + \left(\sum_{i=1}^{hk} c_{hk} e_{i,hk} \right) x^{hk-1} \\
&= c_1 \sum_{j=1}^{hk} e_{1,j} x^{j-1} + c_2 \sum_{j=1}^{hk} e_{2,j} x^{j-1} + \ldots + c_{hk} \sum_{j=1}^{hk} e_{hk,j} x^{j-1}.
\end{aligned} \tag{4.6}$$

So $\|b_1\|$ is "almost" an upper bound for the polynomial $r(x)$ in the entire range $|x| \leq X$. Notice also that $r(x_0) = 0 \pmod{N^{h-1}}$ since each sum is zero modulo N^{h-1}.

Combining equations 4.4 and 4.5 means that, from making the matrix M with a natural number X, we can form a polynomial $r(x)$ that satisfies $r(x_0) = 0 \pmod{N^{h-1}}$ and

$$|r(x)| \leq \left(2^{(hk-1)/4} \sqrt{hk} \right) X^{(hk-1)/2} N^{(h-1)/2} \quad \text{for all } |x| \leq X.$$

Thus choosing

$$X = \left\lceil \left(2^{-1/2} (hk)^{-1/(hk-1)} \right) N^{(h-1)/(hk-1)} \right\rceil - 1 \tag{4.7}$$

means that we can form a polynomial $r(x)$ such that $r(x_0) = 0 \pmod{N^{h-1}}$ and

$$|r(x)| < N^{h-1} \quad \text{for all} \quad |x| \leq X.$$

This implies that $r(x_0) = 0$ over the integers as well, for any x_0 such that $|x_0| \leq X$, and $p(x_0) = 0 \pmod{N}$. Solving this univariate equation over the integers can be done in polynomial time (for instance by Hensel lifting the linear factors, or by finding small factors of the trailing coefficient), and then one can test each solution to see if it satisfies $p(x_0) = 0 \pmod{N}$. Notice that the bound $X \to 2^{-1/2}N^{1/k}$ as $h \to \infty$.

The polynomial $r(x)$ can be formed from equation 4.6 or the coefficients may be obtained by dividing the entries of the vector b_1 by appropriate powers of X.

This kind of reasoning has appeared before in (Hastad, 1988).

4.2 A review of Coppersmith's method

Below we outline the approach given in (Coppersmith, 1996b) for finding small roots of univariate modular equations. One firstly chooses a natural number X, and forms the lower triangular $(2hk - k) \times (2hk - k)$ matrix

$$M = \left(\begin{array}{c|c} D & A \\ \hline O_{hk} & D' \end{array} \right),$$

where

- $D = (d_{i,j})$ is an $(hk \times hk)$ diagonal matrix with entries $d_{i,i} = X^{1-i}$,
- $A = (a_{i,j})$ is an $(hk \times (h-1)k)$ matrix, where the entry $a_{i,j}$ is the coefficient of x^i in the expression $x^u((p(x))^v$, with $v = \lfloor (k+j-1)/k \rfloor$, and $u = (j-1) - k(v-1)$.
- $D' = (d'_{i,j})$ is an $((h-1)k \times (h-1)k)$ diagonal matrix with entries $d'_{i,i} = N^v$ where $v = \lfloor (k+i-1)/k \rfloor$.

This has determinant

$$\det(M) = N^{hk(h-1)/2} X^{-hk(hk-1)/2}. \tag{4.8}$$

Since there is a triangular sub-matrix of A with 1's on the diagonal it is possible to transform the matrix M (using integral elementary row operations implied by a matrix

$H_1 \in GL_n(\mathbb{Z})$), to

$$\tilde{M} = H_1 M = \left(\begin{array}{c|c} \hat{M} & 0_{(hk\times(h-1)k)} \\ \hline A' & I_{(h-1)k} \end{array} \right).$$

This means that the absolute value of the determinant of both $\tilde{M}$ and $\hat{M}$ are the same as that given by equation 4.8. We then reduce $\hat{M}$ using lattice basis reduction to give a matrix $B = H_2\hat{M}$. Let B^* (with row vectors b_i^*) denote this basis after Gram-Schmidt orthogonalisation. We know from equation 3.15 that

$$\|b_{hk}^*\| \geq 2^{-(hk-1)/4} N^{(h-1)/2} X^{-(hk-1)/2}. \tag{4.9}$$

Assume that $p(x_0) = 0 \pmod N$ and let $y_0 = p(x_0)/N \in \mathbb{Z}$. Define the following vector of length $(2hk-k)$,

$$c_0 = \left(1, x_0, \ldots, x_0^{hk-1}, -y_0, -y_0x_0, \ldots, -y_0x_0^{k-1}, -y_0^2, \ldots, -y_0^{h-1}x_0^{k-1}\right). \tag{4.10}$$

Further, when given a vector v of length $(2hk-k)$ that has 0's for the last $(h-1)k$ entries, then denote by $[v]_{\mathrm{sh}}$ the vector "shortened" to one of length (hk).

Assuming that $|x_0| \leq X$, the above implies

$$\begin{aligned}
\sqrt{hk} &\geq \left\|\left(1, x_0/X, \ldots (x_0/X)^{hk-1}, 0, \ldots, 0\right)\right\| \\
&= \|c_0 M\| \\
&= \|c_0 H_1^{-1}\tilde{M}\| \\
&= \|[c_0H_1^{-1}]_{\mathrm{sh}}\hat{M}\| \quad \text{since } (c_0H_1^{-1})_i = 0 \text{ for } i > hk \\
&= \|[c_0H_1^{-1}]_{\mathrm{sh}}H_2^{-1}B\| \\
&= \|c'B\| \quad \text{where } c' = [c_0H_1^{-1}]_{\mathrm{sh}}H_2^{-1} \\
&= \|c''B^*\| \quad \text{for some } c'' \in R^{hk} \\
&\geq \|c''_{hk}b^*_{hk}\| \\
&= \|c'_{hk}b^*_{hk}\| \quad \text{since } c''_{hk} = c'_{hk} \in \mathbb{Z} \\
&= |c'_{hk}|\,\|b^*_{hk}\| \\
&\geq |c'_{hk}|2^{-(hk-1)/4}N^{(h-1)/2}X^{-(hk-1)/2},
\end{aligned} \tag{4.11}$$

which means, since $c'_{hk} \in \mathbb{Z}$, that $c'_{hk} = 0$ for any

$$X < \left(2^{-1/2}(hk)^{-1/(hk-1)}\right) N^{(h-1)/(hk-1)}. \tag{4.12}$$

If instead of c_0 we consider the variable vector

$$\begin{aligned} c(x) &= \Big(1, x, \ldots, x^{hk-1}, -p(x)/N, -xp(x)/N, \ldots, -x^{k-1}p(x)/N, \\ &\quad -(p(x)/N)^2, -x(p(x)/N)^2, \ldots, -x^{k-1}(p(x)/N)^{h-1}\Big), \end{aligned} \tag{4.13}$$

which satisfies $c(x_0) = c_0$, then $c'_{hk}(x)$ is a univariate polynomial given by

$$c'_{hk}(x) = [c(x)H_1^{-1}]_{\text{sh}} \cdot ((H_2^{-1})^t)_{hk}. \tag{4.14}$$

This has integer coefficients after multiplying through by N^{h-1}, and with X chosen as large as possible (from equation 4.12), this polynomial must satisfy $c'_{hk}(x_0) = 0$ for any $|x_0| < X$.

The polynomial $c'_{hk}(x)$ is not identically zero since it is the sum of integer multiples of polynomials of different degrees, and not all the multiples can be zero; otherwise H_2 would have zero determinant.

4.3 Examples

We examine the approach used by both methods to solve the equation $p(x) = x^2 + 14x + 19 = 0 \pmod{35}$ with $h = 3$ (thus $X = 2$). Actually this polynomial has a solution $x_0 = 3$, but as we will see the methods still find it even though $x_0 > X$. It is often the case that the theoretical X given in the previous two sections is a little too low.

4.3.1 Coppersmith's method

Coppersmith's method would firstly form the $(10) \times (10)$ matrix below.

$$M = \begin{pmatrix}
1 & & & 0 & & & 19 & 0 & 361 & 0 \\
 & 2^{-1} & & & & & 14 & 19 & 532 & 361 \\
 & & 2^{-2} & & & & 1 & 14 & 234 & 532 \\
 & & & 2^{-3} & & & & 1 & 28 & 234 \\
 & & & & 2^{-4} & & & & 1 & 28 \\
 & & & & & 2^{-5} & & & & 1 \\
 & & & & & & 35 & & & \\
 & & 0 & & & & & 35 & & \\
 & & & & & & & & 1225 & \\
 & & & & & & & & & 1225
\end{pmatrix}$$

This is transformed (using elementary row operations) to

$$\tilde{M} = \begin{pmatrix}
1 & 0 & -19 \times 2^{-2} & 266 \times 2^{-3} & -3363 \times 2^{-4} & 42028 \times 2^{-5} & & & & \\
 & 2^{-1} & -14 \times 2^{-2} & 177 \times 2^{-3} & -2212 \times 2^{-4} & 27605 \times 2^{-5} & & & & \\
 & & -35 \times 2^{-2} & 490 \times 2^{-3} & -5530 \times 2^{-4} & 58800 \times 2^{-5} & & 0 & & \\
 & & & -35 \times 2^{-3} & 980 \times 2^{-4} & -19250 \times 2^{-5} & & & & \\
 & 0 & & & -1225 \times 2^{-4} & 34300 \times 2^{-5} & & & & \\
 & & & & & -1225 \times 2^{-5} & & & & \\
 & & 2^{-2} & -14 \times 2^{-3} & 158 \times 2^{-4} & -1680 \times 2^{-5} & 1 & & & \\
 & & & 2^{-3} & -28 \times 2^{-4} & 550 \times 2^{-5} & & 1 & & \\
 & 0 & & & 2^{-4} & -28 \times 2^{-5} & & & 1 & \\
 & & & & & 2^{-5} & & & & 1
\end{pmatrix}$$

where $\tilde{M} = H_1 M$, and

$$H_1 = \begin{pmatrix} 1 & 0 & -19 & 266 & -3363 & 42048 & & & & \\ & 1 & -14 & 177 & -2212 & 27605 & & & & \\ & & -35 & 490 & -5530 & 58800 & 1 & & & \\ & & & -35 & 980 & -19250 & & 1 & & \\ & & & & -1225 & 34300 & & & 1 & \\ & & & & & -1225 & & & & 1 \\ & & 1 & -14 & 158 & -1680 & & & & \\ & & & 1 & -28 & 550 & & & & \\ & & & & 1 & -28 & & & & \\ & & & & & 1 & & & & \end{pmatrix}.$$

We then examine (after clearing denominators and swapping the rows and columns), the $(6) \times (6)$ sub-matrix below.

$$\hat{M} = \begin{pmatrix} -1225 & & & 0 & & \\ 34300 & -1225 \times 2 & & & & \\ -19250 & 980 \times 2 & -35 \times 2^2 & & & \\ 58800 & -5530 \times 2 & 490 \times 2^2 & -35 \times 2^3 & & \\ 27605 & -2212 \times 2 & 177 \times 2^2 & -14 \times 2^3 & 2^4 & \\ 42048 & -3363 \times 2 & 266 \times 2^2 & -19 \times 2^3 & 0 & 2^5 \end{pmatrix}$$

This is LLL reduced to

$$B_2 = \begin{pmatrix} 100 & 0 & 15 \times 2^2 & 0 & -10 \times 2^4 & 0 \\ -51 & 36 \times 2 & 20 \times 2^2 & -11 \times 2^3 & -4 \times 2^4 & -2 \times 2^5 \\ 108 & -45 \times 2 & 33 \times 2^2 & 2 \times 2^3 & 6 \times 2^4 & 1 \times 2^5 \\ 16 & -90 \times 2 & 1 \times 2^2 & 4 \times 2^3 & -3 \times 2^4 & 2 \times 2^5 \\ 13 & -58 \times 2 & -4 \times 2^2 & 6 \times 2^3 & 5 \times 2^4 & -4 \times 2^5 \\ 154 & -43 \times 2 & -28 \times 2^2 & -25 \times 2^3 & 7 \times 2^4 & -2 \times 2^5 \end{pmatrix},$$

where $B_2 = H_2 \hat{M}$, and

$$H_2 = \begin{pmatrix} 0 & 4 & 5 & 4 & -10 & 0 \\ 3 & 4 & 6 & 3 & -4 & -2 \\ -8 & -4 & -5 & -3 & 6 & 1 \\ 1 & 0 & 0 & 0 & -3 & 2 \\ -2 & -2 & -5 & 0 & 5 & -4 \\ 15 & 3 & 7 & -1 & 7 & -2 \end{pmatrix}.$$

Now

$$H_1^{-1} = \begin{pmatrix} 1 & & & & & & 19 & 0 & 361 & 0 \\ & 1 & & & & & 14 & 19 & 532 & 361 \\ & & & & & & 1 & 14 & 234 & 532 \\ & & & 0 & & & & 1 & 28 & 234 \\ & & & & & & & & 1 & 28 \\ & & & & & & & & & 1 \\ & & 1 & & & & 35 & & & \\ & & & 1 & & & & 35 & & \\ & & & & 1 & & & & 1225 & \\ & & & & & 1 & & & & 1225 \end{pmatrix},$$

so $[c(x)H_1^{-1}]_{\mathrm{sh}}$ has (as is typical) the following form

$$\left(1, x, \frac{-(x^2+14x+19)}{35}, \frac{-x(x^2+14x+19)}{35}, \frac{-(x^2+14x+19)^2}{1225}, \frac{-x(x^2+14x+19)^2}{1225}\right),$$

and

$$H_2^{-1} = \begin{pmatrix} -5 & 4 & -2 & 1 & -1 & -2 \\ 138 & -109 & 56 & -18 & 31 & 57 \\ -77 & 60 & -32 & 8 & -18 & -32 \\ 231 & -171 & 104 & -7 & 59 & 98 \\ 109 & -82 & 48 & -6 & 27 & 46 \\ 166 & -125 & 73 & -9 & 41 & 70 \end{pmatrix}.$$

Thus taking the dot product of $[c(x)H_1^{-1}]_{\mathrm{sh}}$ with the last column of H_2^{-1} (and then multiplying by 1225) gives the polynomial

$$r(x) \quad = \quad 2x^5 - x^4 - 8x^3 - 24x^2 + 8x + 3,$$

which evaluates to zero over the integers at the root of $(p(x) \pmod{35})$, $x_0 = 3$.

4.3.2 The alternative method

The approach given in Section 4.1 would immediately form the $(6) \times (6)$ matrix below.

$$M_1 = \begin{pmatrix} 1225 & & & & 0 & \\ 0 & 1225 \times 2 & & & & \\ 665 & 490 \times 2 & 35 \times 2^2 & & & \\ 0 & 665 \times 2 & 490 \times 2^2 & 35 \times 2^3 & & \\ 361 & 532 \times 2 & 234 \times 2^2 & 28 \times 2^3 & 2^4 & \\ 0 & 361 \times 2 & 532 \times 2^2 & 234 \times 2^3 & 28 \times 2^4 & 2^5 \end{pmatrix}$$

This is then LLL reduced to

$$B_1 = \begin{pmatrix} 3 & 8 \times 2 & -24 \times 2^2 & -8 \times 2^3 & -1 \times 2^4 & 2 \times 2^5 \\ 49 & 50 \times 2 & 0 & 20 \times 2^3 & 0 & 2 \times 2^5 \\ 115 & -83 \times 2 & 4 \times 2^2 & 13 \times 2^3 & 6 \times 2^4 & 2 \times 2^5 \\ 61 & 16 \times 2 & 37 \times 2^2 & -16 \times 2^3 & 3 \times 2^4 & 4 \times 2^5 \\ 21 & -37 \times 2 & -14 \times 2^2 & 2 \times 2^3 & 14 \times 2^4 & -4 \times 2^5 \\ -201 & 4 \times 2 & 33 \times 2^2 & -4 \times 2^3 & -3 \times 2^4 & 1 \times 2^5 \end{pmatrix},$$

where $B_1 = HM_1$, and

$$H = \begin{pmatrix} 70 & 46 & -98 & 32 & -57 & 2 \\ 73 & 48 & -104 & 32 & -56 & 2 \\ 55 & 36 & -74 & 27 & -50 & 2 \\ 125 & 82 & -171 & 60 & -109 & 4 \\ -175 & -115 & 254 & -74 & 126 & -4 \\ 41 & 27 & -59 & 18 & -31 & 1 \end{pmatrix}.$$

The polynomial relationship required can be obtained from

- looking at the vector b_1, and forming the coefficients by dividing the entries by $1, 2, \ldots 2^5$; this gives the polynomial $r(x) = 2x^5 - x^4 - 8x^3 - 24x^2 + 8x + 3$,
- using the entries of $h_1 = (\alpha_i)$ to form the sum

$$r(x) = \alpha_1 N^2 + \alpha_2 N^2 x + \alpha_3 N p(x) + \alpha_4 N x p(x) + \alpha_5 p^2(x) + \alpha_6 x p^2(x),$$

  which is (obviously) the same polynomial as above.

4.3.3 A graphical explanation of the new methods

The algorithm in Section 4.1 can be seen from a pictorial viewpoint. Given the univariate modular equation $p(x) = x^2 + 14x + 19 \pmod{35}$ we can represent this by the following diagrams where Figure 1.1 gives the value of $p(x)$ modulo 35 in the range $-17 \ldots 18$ for $-17 \leq x \leq 18$, and Figure 4.3.1(B) gives the multiple of 35 that needed to be subtracted. The sharp vertical lines in Figure 4.3.1(A) do not really exist, but serve to show discontinuity. The problem is to determine whether or not any of the points near the origin actually lie on the $r = 0$ axis.

Figure 4.3.1 The polynomial $x^2 + 14x + 19 = 35q + r$, for $-17 \leq x \leq 18$

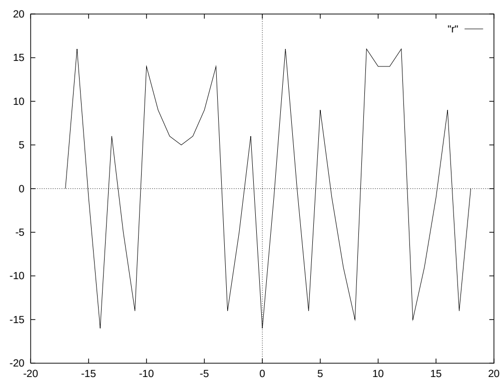


Figure A: r vs. x

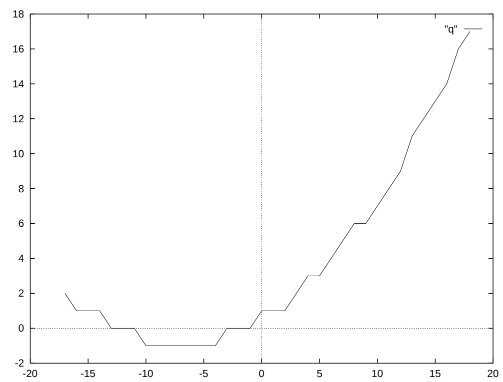


Figure B: q vs. x

The technique described in Section 4.1 finds a multiple of $p(x)$ modulo N^{h-1} that is itself small for small values of x (and obviously shares the same roots as $p(x)$). In our example

$$\begin{aligned} r(x) &= m(x)p(x) \pmod{N^2} \\ &= (2x^3 - 29x^2 + 360x - 4513)(x^2 + 14x + 19) \pmod{1225} \\ &= 2x^5 - x^4 - 8x^3 - 24x^2 + 8x + 3 \pmod{1225}, \end{aligned}$$

and we represent $r(x)$ graphically, in the region $-17 \leq x \leq 18$, below.

Figure 4.3.2 The polynomial $r(x) = 35q + r$, for $-17 \leq x \leq 18$

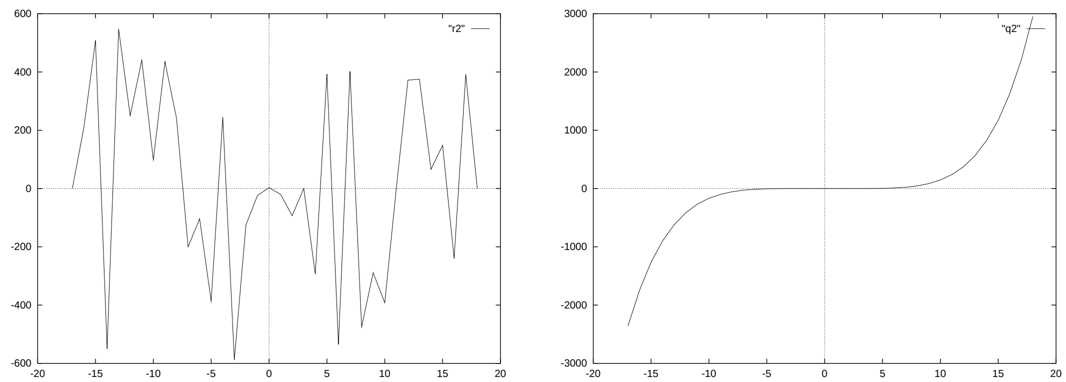


Figure A: r vs. x **Figure B:** q vs. x

It can be seen that no multiples of 1225 need to be removed in the region $-3 \leq x \leq 3$, which means that the polynomial is true over the integers in this area (shown by the "continuity" of r). Solving this equation will determine whether $r(x)$ touches the axis at an integral point.

The process described in (Vallée *et al.*, 1988) can be thought of as multiplying $p(x)$ by a suitably chosen *constant* and reduce modulo N, whereas the approaches given in this chapter multiply $p(x)$ by a *polynomial* of degree $(h-1)k-1$ and reducing modulo N^h.

4.4 The connection between the methods

We must actually show that it is the theory in Section 3.4 that links the lattices produced by the two methods given in Sections 4.1 and 4.2.

Define the $(hk) \times (hk)$ matrix,

$$E = \left(\begin{array}{c|c} \begin{matrix} I_k \\ 0_{(h-1)k} \end{matrix} & A \end{array} \right),$$

where A is defined as in Section 4.2. By the process also defined in Section 4.2, the matrix $\hat{M}$ is almost E^{-1}, but the j'th column is multiplied by X^{1-j}, and the i'th row is multiplied by $-N^v$, where $v = \lfloor (j-1)/k \rfloor$. Alternatively stated, $\hat{M} = PE^{-1}Q$, where $P = (p_{i,j})$ is diagonal and has entries $p_{i,i} = -N^v$ (v defined above), and $Q = \operatorname{diag}\{1, X^{-1}, \ldots, X^{-(hk-1)}\}$.

This implies that $\hat{M}^{-t} = P^{-t}E^tQ^{-t}$, with $P^{-t} = (p'_{i,j})$ diagonal and such that $p'_{i,i} =$

$-N^{-v}$, and $Q^{-t} = \text{diag}\ \{1, X, \ldots, X^{hk-1}\}$. After clearing denominators we verify that $N^{h-1}\hat{M}^{-t} = M_1$, where M_1 is the matrix formed by the method given in Section 4.1.

4.5 Slight improvements

There are (at least) two relatively small improvements that can be made; the first decreases the size of the initial matrix by one, and the second considers placing rows corresponding to different polynomials into the initial matrix.

Before examining these slight improvements it is interesting to note that it is not always better to perform each of the steps described in Section 4.1. For instance if one considers the (relatively sparse) cubic polynomial $x^3 + ax + b = 0 \pmod N$, then we might firstly reduce the matrix on the left, and then "improve" this to the matrix on the right.

$$\begin{pmatrix} N & 0 & 0 \\ 0 & NX & 0 \\ b & aX & X^3 \end{pmatrix} \qquad \begin{pmatrix} N & 0 & 0 & 0 & 0 \\ 0 & NX & 0 & 0 & 0 \\ 0 & 0 & NX^2 & 0 & 0 \\ b & aX & 0 & X^3 & 0 \\ 0 & bX & aX^2 & 0 & X^3 \end{pmatrix}$$

However it can easily be checked that the bound on X implied by the matrix on the left is better!

4.5.1 Removing the constant column

Consider removing from the matrix M the column corresponding to the constant terms of the polynomials (i.e. the left hand one), and the row corresponding to the constant polynomial N^{h-1} (i.e. the top one), and then dividing all the entries by X to form a matrix M'. If one now applies the LLL algorithm to this basis we find a small vector $b'_1 = h'_1 M'$ that satisfies

$$||b'_1||_2 \quad \leq \quad 2^{(hk-2)/4} X^{(hk-2)/2} N^{((hk-2)/(hk-1))((h-1)/2)}$$

Let $h_1 = (e, \alpha_1, \ldots, \alpha_{hk-1})$ where $h'_1 = (\alpha_i)$ and e is such that $|(h_1 M)_1| \leq (1/2)N^{h-1}$ (calculated by reducing $(h'_1 M')_1$ modulo N^{h-1}), and then consider the vector $b_1 =$

$h_1 M$. This has $\|\cdot\|_1$ norm

$$\begin{aligned}\|b_1\|_1 &\leq |(h_1 M)_1| + \|b_1'\|_1 X \\ &\leq (1/2)N^{h-1} + \sqrt{(hk-1)}\,\|b_1'\|_2 X \\ &\leq (1/2)N^{h-1} + \sqrt{(hk-1)}\,2^{(hk-2)/4} X^{hk/2} N^{\left(\frac{hk-2}{hk-1}\right)\left(\frac{h-1}{2}\right)} \\ &< N^{h-1},\end{aligned}$$

whenever

$$X \quad < \quad \left(2^{-\frac{hk+2}{2hk}}(hk-1)^{-1/hk}\right) N^{\left(\frac{h-1}{hk-1}\right)}.$$

Thus this improvement actually marginally increases the permissible bound X whilst reducing the size of the initial matrix!

4.5.2 Including different polynomials

This improvement comes from the following observation.

Lemma 4.5.1 *Given a polynomial* $p(x)$ *modulo* N *of degree* k*, and provided* $\gcd(N, k!) = 1$*, then one can produce a polynomial* $q(x)$ *modulo* $(k!)N$ *also of degree* k*, which shares the same roots as* $p(x)$*.*

Proof: We know $p(x_0) = \sum_{i=0}^{k} p_i x^i = 0 \pmod N$, and $q(x) = \prod_{i=1}^{k}(x-i) = \sum_{i=0}^{k} q_i x^i = 0 \pmod{k!}$ for all $x \in \mathbb{Z}$. Therefore, using the Chinese remainder theorem, we can produce a polynomial $r(x) = \sum_{i=0}^{k} r_i x^i$ where $r_i \equiv p_i \pmod N$ and $r_i \equiv q_i \pmod{k!}$. By its formation $r(x)$ satisfies $r(x_0) = 0 \pmod N$, and $r(x_0) = 0 \pmod{k!}$, so $r(x_0) = 0 \pmod{(k!)N}$. $\square$

This theorem may be immediately applied to the polynomial $p(x)$ in question, slightly increasing N (and hence X), but it is better to make use of the theorem continually whilst creating the matrix M, i.e. making all the polynomials in the matrix zero modulo $(hk-1)!N^{h-1}$.

Another approach might be to use the fact that $(p(x))^m = 0 \pmod{N^m}$, and thus we can find a polynomial $r(x) = 0 \pmod{(mk!)N^m}$ with the same roots as $p(x)$, and we can find all solutions of this up to $O((mk!)^{1/mk} N^{1/k})$ in polynomial time. However $(n!)^{1/n} < n$ for all $n \geq 2$, so this is not considered a large improvement, and probably completely outweighed by the increase in the size of the associated matrix.

4.6 Implementations and practical results

Implementations of both algorithms have been written by the author in `C` using `Gnu MP` as a multi-precision integer package. The timing results are from runs on a SGI Indy with one 100MHz IP22 processor. The main part of the program was an efficient implementation of the integral LLL algorithm, details of which may be found in (Cohen, 1991).

Firstly let it be said that these algorithms only find solutions of univariate modular equations up to $O(N^{1/k})$, and that the time to find these solutions is of complexity

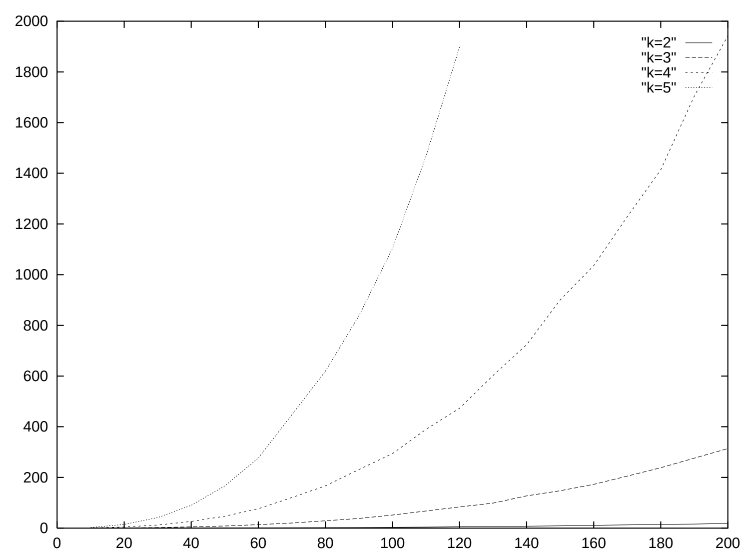
$$t \;=\; O(h^9 k^6 \log^3 N). \qquad (4.15)$$

Therefore as the degree of the polynomial k increases, less possible solutions are checked in greater time, i.e. the method becomes increasingly bad compared to a brute force search.

Several timing graphs are given below for various h, k and N. For a given degree k these are created by forming 3 pseudo-random polynomials of degree k, which have a maximal X as a root, and then averaging the time taken by the algorithm on these polynomials.

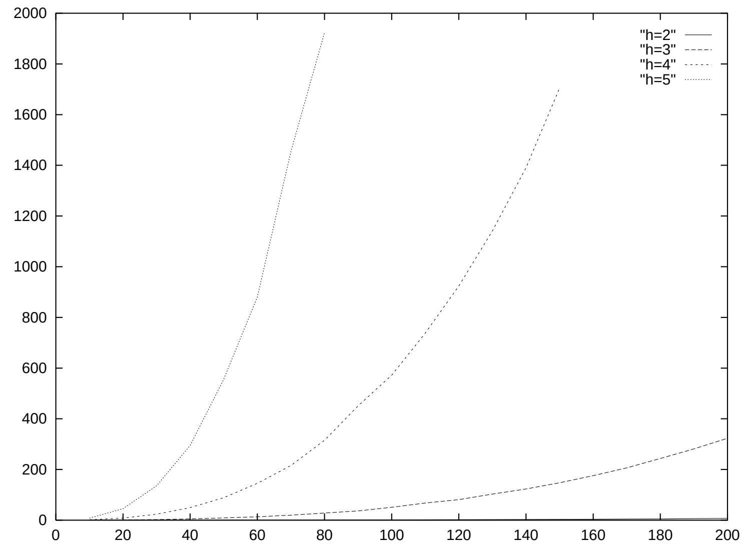
Below we show the average times needed to find solutions up to $O(N^{2/(3k-1)})$ (i.e. $h = 3$) for polynomials modulo 10^n, $10 \leq n \leq 200$ of degrees 2,3,4 and 5.

Figure 4.6.1 A plot of time t (in seconds) vs. $\log_{10} N$ with $h = 3$, for various k



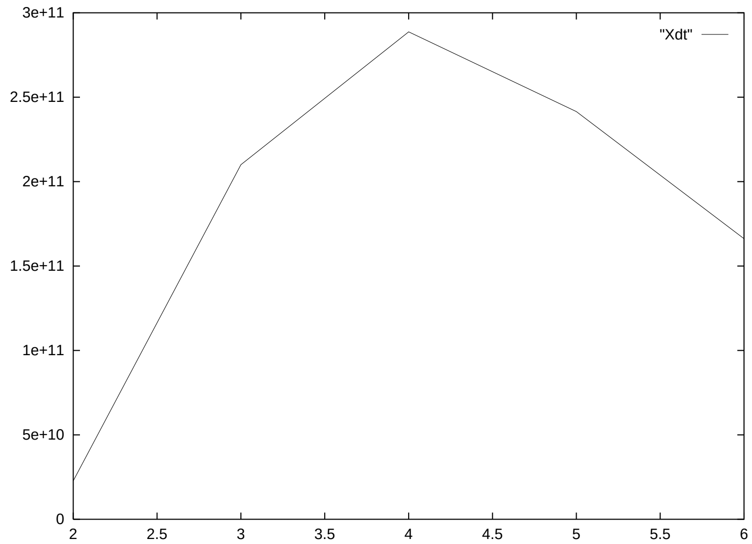
As we increase h (to increase X), this also increases the time needed to find these solutions. Below we show the average times to find solutions up to $O(N^{(h-1)/(3h-1)})$ for cubic polynomials modulo 10^n, $10 \leq n \leq 200$ with $h = 2, 3, 4$ and 5.

Figure 4.6.2 A plot of time t (in seconds) vs. $\log_{10} N$ for cubic polynomials with $2 \leq h \leq 5$



Comparing Figures 4.6.1 and 4.6.2 (or examining the complexity given by equation 4.15) we see that the algorithm is far more sensitive to an increase in h than one in k. Furthermore $X \to O(N^{1/k})$ as $h \to \infty$ (i.e. $t \to \infty$), which means there must be a compromise as to which h to use to maximise the number of X checked per unit time. This is shown below for cubic polynomials modulo $N = 10^{50}$.

Figure 4.6.3 An optimum h for X/t for cubic polynomials modulo $N = 10^{50}$



All the above results have been achieved using the new algorithm with an in-built early exit, and the algorithmic improvement given in Section 4.5.1 (but not that explained in Section 4.5.2; the effect of which is yet to be fully analysed). To see the effect of *not* using the improvement in Section 4.5.1 we show the time taken for the two cases $(h = 2, k = 2)$ and $(h = 4, k = 4)$ for various N.

Figure 4.6.4 The effect of removing the constant column

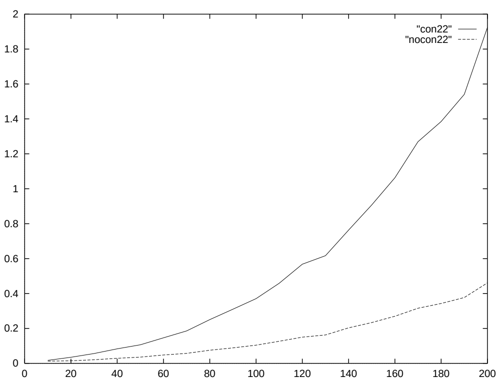


Figure A:
Time, t vs. $\log_{10} N$ for $h = 2$, $k = 2$

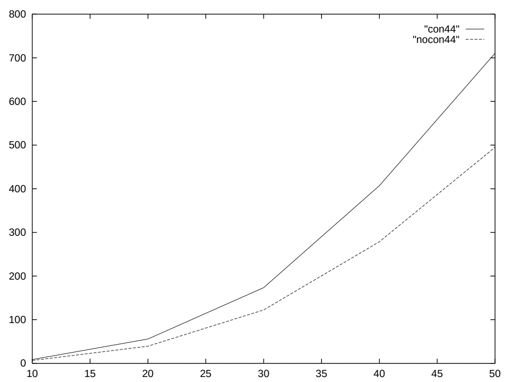


Figure B:
Time, t vs. $\log_{10} N$ for $h = 4$, $k = 4$

Lastly we examine the effect of the early exit of the LLL algorithm, by comparing the algorithm given in Section 4.2 to that given in Section 4.1 where no other improvements are used. The figure below shows the two cases $(h = 3, k = 3)$, and $(h = 3, k = 4)$. The main speed up seems to come when there is a root that is significantly less than the maximum X, enabling the last row(s) not to be used in finding a small enough element. This is something that only the new method is able to take advantage of, and the cases below indicate that the theoretical maximum X is a little too low.

Figure 4.6.5 The effect of an early exit

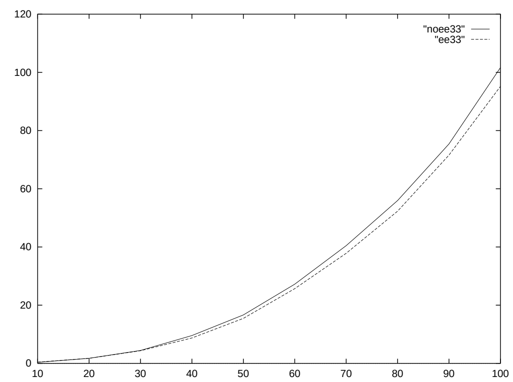


Figure A:
Time, t vs. $\log_{10} N$ for $h = 3$, $k = 3$

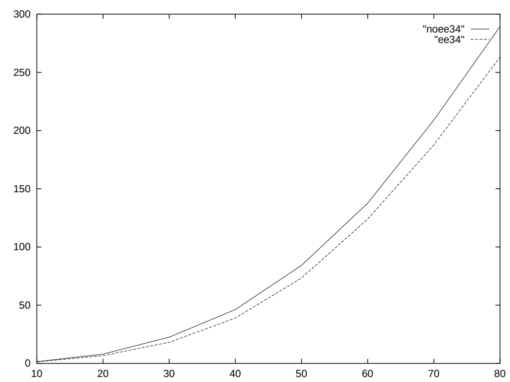


Figure B:
Time, t vs. $\log_{10} N$ for $h = 3$, $k = 4$

Since the publication of the dual method further practical results have been achieved in (Coupé *et al.*, 1999) which make use of the NTL library of Victor Shoup (see (Shoup, 1995)) – this is a far more efficient and sophisticated implementation of the lattice reduction algorithm of Section 3.3. However their results do confirm the above analysis.

4.7 Algebraic univariate modular equations and general multivariate equations

In this section we briefly discuss the general problem of finding small solutions to multivariate modular equations, and then develop techniques using algebraic numbers that allow us to attack certain instances of this problem. This work was undertaken at IBM, Yorktown Heights and the helpful input of Don Coppersmith is gratefully acknowledged.

Some discussion was given in (Coppersmith, 1996a) in which it was explained that the general lattice techniques (e.g. as described in Section 4.1) can be applied to multivariate modular equations. We omit the exact details here, but the general approach is that the rows of the matrix now correspond to the (multivariate) monomials of the relevant modular polynomials. It was explained that there are two problems with this extension: firstly one needs to ensure that the lattice reduction techniques produce enough equations that are true over the integers (i.e. produce enough rows with small enough norm), and secondly that the resulting equations are *algebraically* independent, so that resultant or Gröbner base techniques will recover the solutions. As noted in (Coppersmith, 1996a) the success of such an approach must be limited in general because it is shown in (Manders & Adleman, 1978) that finding bounded solutions even to the trivariate integer equation $x^2 - z = 0 \bmod N$, is NP-hard.

The first of the above problems (ensuring enough small vectors) was studied in (Jutla, 1998). If we label the vectors produced by the LLL reduction algorithm by $b_1, \ldots, b_n$, then equation 3.14 gives an upper bound for the size of b_1. Jutla realised that a lower bound for any lattice element is achieved by the minimum (in terms of absolute value) entry on the diagonal of the Coppersmith matrix. He then used this to bound the vectors b_i for $i \geq 2$. This idea is generalised by Lemma 3.2.5 to produce the improved bounds on the b_i given by equation 3.19.

A problem with this extension of Coppersmith's algorithm is that the number of monomials (hence dimension of the lattice) can increase dramatically when exponentiating the initial polynomial (see Section 4.7.4 for an example).

In the remainder of this section we concentrate on a particular type of multivariate modular equation, namely solving univariate polynomials in an algebraic variable, "modulo" an algebraic ideal. In particular we attack low exponent RSA (with encrypting exponent e) with random padding in more than one location, i.e. we assume we know most of the message M, but do not know some small random pads x_i. However the locations of these pads throughout the message (the F_i shifts below - which are perhaps powers of 2) are known. This implies we must solve the following equation:

$$\left(M + \sum_{i=0}^{m-1} F_i x_i\right)^e - c \quad = \quad 0 \bmod N. \qquad (4.16)$$

We will further assume that $F_i = F^i$ for some F, i.e. that the padding is spread in equally spaced blocks throughout the message, and also that there exists a polynomial r, of low degree m, and with small coefficients (of absolute value at most R) such that

$$\begin{aligned} r(F) &= 0 \bmod N \\ &= kN, \quad \text{and } \gcd(k, N) = 1. \end{aligned}$$

Under these assumptions we can re-write equation 4.16 as the following

$$\left(M + \sum_{i=0}^{m-1} x_i F^i\right)^e - c \quad = \quad 0 \bmod N, \tag{4.17}$$

and then, since $r(F) = 0 \bmod N$, we can introduce the algebraic number α such that $r(\alpha) = 0$. It follows that $(\alpha - F)g(\alpha) = kN$ for some $g(\alpha)$, which suggests that we work modulo the ideal generated (over $\mathbb{Z}[\alpha]$) by N and $\alpha - F$. We shall denote this ideal $I = \langle N, \alpha - F\rangle_{\mathbb{Z}[\alpha]}$, and we shall continue to use the "mod" notation to denote algebraic numbers that are in the same equivalence classes of $\mathbb{Z}[\alpha]/I$.

With the use of the algebraic number α, we may re-write equation 4.17 in the following way:

$$\left(M + \sum_{i=0}^{m-1} x_i \alpha^i\right)^e - c \quad = \quad 0 \bmod I. \tag{4.18}$$

It might be possible to develop a solution to this problem using the work in (Fieker & Pohst, 1996) and (Schiemann, 1998) (see Section 3.6), however extending Theorem 4.0.1 to this framework seems a daunting task, and also the complexity of such an approach may be unclear. Instead we develop techniques based on the ordinary (integral) lattice reduction techniques, which ensures us of a polynomial time attack, and enables us to show that there are provably weak locations to place the random padding.

It seems important to show that this type of algebraic attack does exists, without being overly drawn in to technical details and efficiency issues. However if the attack or general technique turns out to be particularly useful in some practical situations (cryptological or not), then further and deeper analysis may well be warranted.

Since we are using the integral LLL algorithm for reduction we define the size of an element $\beta \in \mathbb{Z}[\alpha]$ to be the maximum (absolute) size of its coefficients when expressed as a polynomial (of degree at most $m-1$) in α, and denote this $\Xi(\beta)$. This quantity if often referred to as the *height* of β in $\mathbb{Z}[\alpha]$. It would perhaps be nicer to use the concept of the algebraic norm $N(\beta)$, but this seems harder to align with LLL.

In order to extend the general technique of Section 4.1 to solving equation 4.18 we attack the following three sub-problems.

- We bound the (Ξ) size of a univariate polynomial when we have bounds on the (Ξ) size of the variable and the coefficients of the polynomial, and also have bounds on the (absolute) size of the coefficients of the minimal polynomial defining α.

- We find a $\mathbb{Z}$-basis for $I = \langle N, \alpha - F\rangle^i_{\mathbb{Z}[\alpha]}$ for $i \geq 1$, and use this to show that some elements of $\mathbb{Z}[\alpha]$ may be considered minimal modulo I, i.e. they are the smallest element (in terms of the Ξ size) of their equivalence class in $\mathbb{Z}[\alpha]/I$.

- We use the previous two results to build a lattice, which when reduced yields an univariate (algebraic) polynomial that is valid over $\mathbb{Q}(\alpha)$ (i.e. no longer modular). Since $\mathbb{Q}(\alpha)$ is a field the number of solutions is bounded by the degree of the polynomial, and is therefore finite. This implies that the (algebraic) roots of the polynomial may be found by performing resultant or Gröbner base calculations on the integer polynomial equations corresponding to the powers of α. See Section 4.7.4 for an example.

4.7.1 Bounding an algebraic polynomial

Using the above notation, we introduce the algebraic number α such that $r(\alpha) = \sum_{i=0}^{m} r_i \alpha_i$ and $|r_i| < R$. Let us assume that we have two algebraic numbers β, γ, such that $\Xi(\beta) \leq B$ and $\Xi(\gamma) \leq C$. We can express the product $\beta\gamma$ as a polynomial in α of degree $2(m-1)$ with coefficients $\leq mBC$. However Ξ is defined only when we have reduced this with respect to r to be of degree $\leq m-1$. Each time we decrement the degree the coefficients can increase by at most a factor of $2R$, so we have that $\Xi(\beta\gamma) \leq m(2R)^{m-1}BC$. Although this is a rather loose bound, it will serve for our purposes. However note that for any given polynomial r one might be able to calculate a better bound than this, or at least form a good bound on average (see Section 4.7.4 for more details).

We can use the above result repeatedly to bound the (Ξ) size of an algebraic polynomial. If we have

$$\delta = \sum_{i=0}^{m-1} \gamma_i \beta^i,$$

with $\Xi(\beta) \leq X$ and $\Xi(\gamma_i) \leq C_i$, then

$$\Xi(\gamma_i \beta^i) \leq \left(m(2R)^{m-1}X\right)^i C_i \tag{4.19}$$

and so

$$\Xi(\delta) \leq \sum_{i=0}^{i-1} \left(m(2R)^{m-1}X\right)^i C_i.$$

4.7.2 An integral basis for the ideal

We wish to find a $\mathbb{Z}$-basis for the ideal $I_i = \langle N, \alpha - F\rangle^i_{\mathbb{Z}[\alpha]}$ for $i \geq 1$. In order to do this we will firstly show that $I_i = J_i$ where $J_i = \langle N^i, (\alpha - F)^i\rangle_{\mathbb{Z}[\alpha]}$, and then find a $\mathbb{Z}$-basis for the latter ideal.

The elements of $I_i = \langle N, \alpha - F\rangle^i_{\mathbb{Z}[\alpha]}$ for $i \geq 1$ are $\{\beta^i \mid \beta \in I_1\}$, and so obviously $J_i \subseteq I_i$. To show that $I_i \subseteq J_i$ we prove the following lemma.

Lemma 4.7.1 *For all* $0 \leq u, v \leq i$, $u + v = i$ *we have*

$$N^u(\alpha - F)^v \in J_i.$$

Proof: By the definition of α we have

$$\begin{aligned} g(\alpha)(\alpha - F) &= kN, \quad \text{so} \\ g(\alpha)^u(\alpha - F)^u &= k^u N^u \\ h(\alpha)(\alpha - F)^u &= lN^u \\ h(\alpha)(\alpha - F)^{u+v} &= lN^u(\alpha - F)^v, \end{aligned} \tag{4.20}$$

where we allow for cancellation of factors of k^u and the coefficients of $h(\alpha)^u$ in equation 4.20.

Let l' be such that $ll' + sN^v = 1$, then

$$\begin{aligned} l'h(\alpha)(\alpha - F)^{u+v} &= (1 - sN^v)N^u(\alpha - F)^v \\ &= N^u(\alpha - F)^v - s(\alpha - F)^v N^{u+v}. \end{aligned}$$

This shows that $N^u(\alpha - F)^v \in J_{u+v}$ for any $0 \leq u, v \leq i$. □

This lemma implies that $(a(\alpha)N + b(\alpha)(\alpha - F))^i \in J_i$ for any $a(\alpha), b(\alpha) \in Z[\alpha]$, i.e. $I_i \subseteq J_i$, so we have that $I_i = J_i$.

To find a basis for J_i we simply write the coefficients of the $\alpha^j N^i$ and $\alpha^j(\alpha - F)^i$ for $0 \leq j \leq m - 1$ as the rows of a matrix, and find the span (a set of $(m - 1)$ vectors) of these $2(m - 1)$ rows (e.g. by putting the matrix in (row) Hermite normal form). One can view this procedure in a larger matrix which also includes rows with the minimal polynomial of α present. An example is shown below.

Example 4.1 *Let* α *be an algebraic number with minimal polynomial* $r(x) = x^3 +$

$2x^2-4x-1$ *over the integers (the fact that r is monic is unimportant in what follows). We have $r(5)=7\times 22$, so*

$$\begin{aligned}(\alpha^2+7\alpha+31)(\alpha-5) &= 7\times 22,\\ (13\alpha^2+69\alpha+139)(\alpha-5)^2 &= 7\times 22^2.\end{aligned}$$

(In general cancellation of factors of k is unlikely, but we show it in this example to highlight that this situation is no harder than typical).

With $22^2=484$ we can find a $\mathbb{Z}$-basis for I_2 by removing the linear dependencies from the following matrix rows.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 484\\ 0 & 0 & 0 & 484 & 0\\ 0 & 0 & 484 & 0 & 0\\ 0 & 0 & 1 & -10 & 25\\ 0 & 1 & -10 & 25 & 0\\ 1 & -10 & 25 & 0 & 0\\ 0 & 1 & 2 & -4 & -1\\ 1 & 2 & -4 & -1 & 0\end{pmatrix} \longrightarrow \begin{pmatrix} 0 & 0 & 0 & 0 & 484\\ 0 & 0 & 0 & 484 & 0\\ 0 & 0 & 484 & 0 & 0\\ 1 & 0 & 0 & 4 & 1511\\ 0 & 1 & 0 & 2 & 1559\\ 0 & 0 & 1 & 4 & 1803\\ 0 & 0 & 0 & 7 & 2583\\ 0 & 0 & 0 & 0 & 3388\end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 0 & 0 & 0 & 35\\ 0 & 1 & 0 & 0 & 337\\ 0 & 0 & 1 & 0 & 327\\ 0 & 0 & 0 & 1 & 369\\ 0 & 0 & 0 & 0 & 484\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\end{pmatrix}$$

We now show that the determinant of the basis that spans J_i is equal to N^i. In general one can see that the larger matrix (with the minimal polynomial of α included) has a Sylvester matrix as its bottom $(m-1)\times(m-1)$ sub-matrix. The resultant $\mathcal{R}$ of $(x-F)^i$ and $r(x)$ is defined to be the determinant of this Sylvester matrix, and has the property that it equals the product of $r(x)$ applied to the roots of $(x-F)^i$, i.e. in our case $\mathcal{R}=r(F)^i=k^iN^i$.

Thus we can put the Sylvester matrix in row Echelon form with diagonal entries that multiply to $(kN)^i$. Since we have the property $h(\alpha)(\alpha-F)^i=lN^i$, then the only diagonal entry that can contain a factor of N is the rightmost one (with respect to the above example), whereas the factors of k^i may be spread amongst the diagonal entries. When we reduce such a basis with α^jN^i for $0\le j\le m-1$ and use the fact that $\gcd(k,N)=1$ we obtain a basis with rightmost diagonal entry N^i and all other diagonal entries equal to 1. Thus the determinant of the $\mathbb{Z}$-basis of I_i is N^i.

In the above example this $\mathbb{Z}$-basis can be read off from the non-zero vectors in the right-hand m columns of the matrix on the right, i.e.

$$\begin{pmatrix} 1 & 0 & 327 \\ 0 & 1 & 369 \\ 0 & 0 & 484 \end{pmatrix},$$

so $I_2 = \{a(\alpha^2 + 327) + b(\alpha + 369) + 484c \ \mid \ a, b, c \in \mathbb{Z}\}$.

If we apply the LLL algorithm to this basis to give a reduced representation:

$$\begin{pmatrix} -6 & 4 & -2 \\ -4 & -3 & 5 \\ -1 & -7 & -6 \end{pmatrix},$$

then from equation 3.13 we see there cannot exists two elements of $\mathcal{I} \in Z[\alpha]/I_2$ which correspond to vectors (in the above sense) with norm less than $2^{-1}\sqrt{56} \approx 3.7$

In general we will assume that the LLL reduction of the basis for I_i has the smallest vector b_1 such that $|b_1| \approx \det(I_i)^{1/m} = N^{i/m}$, so approximately the same bound holds for any element of I_i, and also the bound applies to all the elements of any basis of I_i. This is a plausible assumption on random lattices, though it would be nicer to justify the assumption in more rigorous terms for our particular lattices.

4.7.3 The lattice and general result

If we let $\beta = \sum_{i=0}^{m-1} x_i \alpha^i$, and $|x_i| \leq X$ so that $\Xi(\beta) \leq X$, then we may re-write equation 4.18 as

$$\begin{aligned} p(\beta) &= (M + \beta)^e - c \qquad (4.21) \\ &= 0 \bmod I. \end{aligned}$$

In a similar way to Section 4.1 we consider many equations which are zero modulo $I_h = \langle N, \alpha - F \rangle_{\mathbb{Z}[\alpha]}^h$ for some integer h. In this case however we first work out integral bases for the ideals I_j for $1 \leq j \leq h-1$, and then we know $\gamma p(\beta)^i = 0 \bmod I_{i+j}$ for any $\gamma \in I_j$ since $p(\beta) = 0 \bmod I$. Let $\gamma_{i,j}$ be the i'th element of a basis for I_j; we consider the following polynomials:

$$\begin{gathered}\beta^k\alpha^i p^h(\beta),\\ \beta^k\gamma_{i,1}p^{h-1}(\beta),\\ \beta^k\gamma_{i,2}p^{h-2}(\beta),\\ \vdots\\ \beta^k\gamma_{i,h-1}p(\beta),\\ \beta^k\gamma_{i,h},\end{gathered}$$

for $0 \le i \le m-1$ and $0 \le k \le h-1$.

To place these polynomials within a matrix we have columns corresponding to the "monomials" $\alpha^i\beta^j$ for $0 \le i \le m-1$ and $0 \le j \le e(h+1)-1$. This produces a square matrix L of dimension $em(h+1)$.

As in the ordinary univariate case, we must multiply the columns of the matrix by different amounts according to their exponent of β. For $0 \le j \le e(h+1)-1$ we multiply the m columns that correspond to β^j by B^j where $B = m(2R)^{m-1}X$ is taken from equation 4.19.

The determinant of the matrix is thus given by

$$\det L \quad = \quad B^{me(h+1)(e(h+1)-1)/2}N^{eh(h+1)/2}.$$

We now LLL reduce the matrix L, and let b_1 be the first vector of the LLL reduced basis. Under the assumption that the smallest element in I_h has Ξ size roughly equal to $N^{h/m}$, and using the fact that the Ξ size of the (algebraic) polynomial corresponding to the b_1 is (roughly) bounded by the norm of this vector, means that if we ensure

$$\left(B^{me(h+1)(e(h+1)-1)/2}N^{eh(h+1)/2}\right)^{1/(em(h+1))} \quad < \quad N^{h/m},$$

then the polynomial (in β) corresponding to b_1 will be equal to zero over $\mathbb{Q}(\alpha)$. By re-substituting $\beta = \sum_{i=0}^{m-1} x_i\alpha^i$ we can then solve the m integer polynomial equations in m variables x_i by using resultant or Gröbner base techniques, since we are assured of a finite number of solutions by virtue of the fact that $\mathbb{Q}(\alpha)$ is a field.

The bound on B implied by equation 4.22 is

$$B \quad < \quad N^{(1/em)(eh/(eh+e-1))},$$

which means that we can find the variables x_i in polynomial time whenever $|x_i| \le X$, and

$$X \quad < \quad N^{(1/(em))(eh/(eh+e-1))}/(m(2R)^{m-1}).$$

We state this result as a general theorem now.

Theorem 4.7.2 *Suppose that* $(M + \sum_{i=0}^{m-1} F^i x_i)^e - c = 0 \bmod N$*, and there exists a polynomial* r *of degree* m *whose coefficients are bounded by* $|r_i| \leq R$*, such that* $r(F) = kN$ *where* $\gcd(k, N) = 1$*. Also assume there exists an* h *such that the smallest (in terms of* Ξ*) element in* $\langle N, \alpha - F\rangle^h_{\mathbb{Z}[\alpha]}$ *is approximately* $N^{h/m}$*. Then the variables* x_i *may be discovered in polynomial time whenever* $|x_i| \leq X$ *and*

$$X \quad < \quad N^{(1/em)(eh/(eh+e-1))}/(m(2R)^{m-1}).$$

Notice that as $h \to \infty$ *the bound* $X \to N^{1/(em)}/(m(2R)^{m-1})$*.*

4.7.4 An example

We now exhibit the method working for the case of

$$N = 2776419924431 \times 11618928225217 = 32259023825026196088576527$$

and $F = 2^{31}$, in which case we find the relation that $F^3 - 3 = 307N$. The fact that we introduce an n'th root of an integer (i.e. $\alpha = \sqrt[3]{3}$), and that we have a relatively small multiple of N (i.e. 307) is not important to the method, but places us in the relatively nice situation that the random padding is (roughly) in n equally spaced blocks starting from the bottom.

Assume we know the plaintext to be $M + F^2 x_2 + F x_1 + x_0$ for some small x_i. We wish to solve $(M+\beta)^3 - c = 0 \bmod \langle N, \alpha - F\rangle_{\mathbb{Z}[\alpha]}$ where $\alpha^3 - 3 = 0$ and $\beta = x_2\alpha^2 + x_1\alpha + x_0$, i.e.

$$p(\beta) \ = \ \beta^3 + 3M\beta^2 + 3M^2\beta + M^3 - c \ = \ 0 \ \bmod \langle N, \alpha - F\rangle_{\mathbb{Z}[\alpha]}.$$

Let us choose $h = 2$. We have that $(\alpha^2 + 2^{31}\alpha + 2^{62})(\alpha - 2^{31}) = -307N$, and the ideals of I_i for $i = 1, 2$ correspond to the rows of the matrices T_i below.

$$T_1 \quad = \quad \begin{pmatrix} 1 & 0 & 32259019213340177661188623 \\ 0 & 1 & 32259023825026193941092879 \\ 0 & 0 & 32259023825026196088576527 \end{pmatrix}$$

$$T_2 = \begin{pmatrix} 1 & 0 & 2312645089043942981849374535652346789883580395254 26 \\ 0 & 1 & 578135898968673335914228437568534195239429376957565 \\ 0 & 0 & 1040644618143607751116024675892568259248385135381729 \end{pmatrix}$$

If we LLL reduce the last of these we obtain

$$\begin{pmatrix} -8029556555425490 7 & 19180902889356093 & 31951861464601304 \\ -16881151744752424 & -63414413809502483 & 82595316698858576 \\ -14581400600148755 & -94876966154403662 & -75696063265047569 \end{pmatrix},$$

and notice that the first row does indeed have norm approximately equal to $0.87N^{2/3}$.

Let u_i, v_i denote the coefficients of β^i in $p(\beta), p^2(\beta)$ respectively, then we wish to reduce the rows of the following matrix

$$\begin{pmatrix} B^8T_0 & B^7v_5T_0 & B^6v_4T_0 & B^5v_3T_0 & B^4v_2T_0 & B^3v_1T_0 & B^2v_0T_0 & & \\ & B^7T_0 & B^6v_5T_0 & B^5v_4T_0 & B^4v_3T_0 & B^3v_2T_0 & B^2v_1T_0 & Bv_0T_0 & \\ & & B^6T_0 & B^5v_5T_0 & B^4v_4T_0 & B^3v_3T_0 & B^2v_2T_0 & Bv_1T_0 & v_0T_0 \\ & & & B^5T_1 & B^4u_2T_1 & B^3u_1T_1 & B^2u_0T_1 & & \\ & & & & B^4T_1 & B^3u_2T_1 & B^2u_1T_1 & Bu_0T_1 & \\ & & & & & B^3T_1 & B^2u_2T_1 & Bu_1T_1 & u_0T_1 \\ & & & & & & B^2T_2 & & \\ & & & & & & & BT_2 & \\ & & & & & & & & T_2 \end{pmatrix},$$

where T_0 is the identity matrix (the T_j serve to show sub-matrices rather than matrix elements), and $B = m(2R)^{m-1}X =$ is to account for the columns corresponding to different powers of β.

The reduction of this type of lattice can be considerably speeded up, by reducing whole blocks (corresponding to the bases of the ideals) together. We omit the details of such an improvement here for the sake of brevity, though details may be found in (Coppersmith & Howgrave-Graham, 1999).

If we compare this to Jutla's approach on $p(x,y,z) = (M + F^2x + Fy + z)^3 - c$, then notice that p^3 has 220 monomials. This corresponds to the dimension of the necessary lattice to reduce, and is almost certainly infeasible with todays computing power.

4.7.5 Conclusions

This attack on RSA encorporates a novel use algebraic numbers in the general lattice techniques. Using these methods we have been able to show that there are provably weak places to hide information even when splitting it in more than one block. The algebraic variant also allows us to significantly reduce the size of the lattice in comparison to treating equation 4.17 simply as a multivariate equation.

At present it seems a relatively theoretical attack in all but a few exceptional situations (e.g. the example in Section 4.7.4). It would be nice if it were possible to extend this technique to more general and practical situations.

One minor extension is to get around the (relatively weak) condition $\gcd(k, N) = 1$: If $1 < \gcd(k, N) < N$ then we have a non-trivial factor of N, and with the application to RSA, this breaks the cryptosystem immediately. If $\gcd(k, N) = N^i$ for some $i \geq 1$ then we can work in the ideal generated by $M = N^i$ and $\alpha - F$, and use the fact that $p(x)^i = 0 \bmod M$.

Finally it would seem that the condition that the minimum Ξ size of any element of $\langle N, \alpha - F\rangle^h_{\mathbb{Z}[\alpha]}$ should be around $N^{h/m}$ may be possible to prove (and so it may be possible to drop the assumption in Theorem 4.7.2). This is the subject of further work.

4.8 Applications to low exponent RSA

The implications of being able to find small solutions of univariate modular equations on the RSA cryptosystem have been well studied, and we start by summing up some well known attacks; for further details see (Boneh, 1999). In Section 4.8.4 we note that the theory of Section 4.7 gives a new result which shows that there are provably weak places to hide information even when splitting this information between many blocks.

As in Section 1.1 we revert to the use of Alice, Bob and Eve. All of these attacks can be prevented by using larger public exponents, but we shall assume that for some applications (perhaps smartcards) this would rather not be done.

4.8.1 One small block of unknown plaintext

The simplest application of Theorem 4.0.1 to RSA is when there is one small block of unknown plaintext. We will explain this idea with Alice using a public exponent of 3 firstly, and then generalise it to an arbitrary exponent.

Suppose Eve knows 2/3 of the message that Bob is sending Alice, but is unsure of the last 1/3, i.e. Eve knows the message M is $(M' + x)$, for some known M' and small $|x| < N^{1/3}$, and she also intercepts the ciphertext c, then all she must do is solve the equation

$$(M' + x)^3 \quad = \quad c \pmod{N},$$

for $|x| < N^{1/3}$, which is possible using the lattice methods.

In general, if Alice uses an encrypting exponent of e, then for this attack to work Eve must know $(e-1)/e$ of Bob's message.

4.8.2 Broadcast attack

It would seem rather unlikely that Eve should know any of Bob's message as in the previous attack. In this section we examine the case when Bob broadcasts the same message M to many of his friends, k say, all with their own moduli N_i and secret keys e_i, as studied in (Hastad, 1988). Let the respective ciphertexts be c_i.

Consider the simplest case that $k = 3$ and all the e_i are 3. If Bob does not try to hide the fact this is happening then Eve will know $M^3 = c_i \pmod{N_i}$ from which she can deduce $M^3 = c \pmod{N_1 N_2 N_3}$ where $c \pmod{N_1 N_2 N_3}$ is obtained by applying the Chinese remainder theorem[2] to the $c_i \pmod{N_i}$. However since M is less than the least N_i and therefore M^3 is less than $N_1 N_2 N_3$ we actually have that $M = c^{1/3}$.

This idea can be generalised even if Bob tries to hide the fact that they are the same M by applying a known polynomial p_i to the message of each friend prior to encryption (e.g. the linear polynomial $p_i(x) = x + 2^m i$ if M were m bits long).

To see this observe that Eve knows $(p_i(M))^{e_i} = c_i \pmod{N_i}$. From this she may find the following

$$p(M) \quad = \quad c \pmod{\prod_{i=1}^{k} N_i}$$

where c is calculated as before, by applying the Chinese remainder theorem to the c_i, and p is also found by applying the Chinese remainder theorem to each of the coefficients of the polynomials $p_i^{e_i}(M)$ (in a similar way to Lemma 4.5.1). This univariate modular equation may obviously be solved using the lattice methods if M is sufficiently small.

[2] In the rare case that the N_i are not relatively prime the CRT would obtain a non-trivial factor of some N_i.

As an example of this, if all the p_i are linear polynomials (c.f. the possible padding function above), then $p_i^{e_i}$ will be of degree e_i, so p will be of degree $e_{\max} = \max\{e_i\}$. Therefore as long as M is less than $\left(\prod_{i=1}^{k} N_i\right)^{1/e_{\max}}$ it will discovered by the lattice methods, and note that this will be the case if $k > e_{\max}$ when all the N_i are approximately the same size.

Note that this attack would fail if Bob were to attach random (rather than known) padding to the message to each friend.

4.8.3 Repeated message and short pad attack

In the last section we saw how random padding might help Bob secure his message, however this is not always the case if there is not enough of it. Consider the situation when Bob sends two messages to Alice that only differ by a small amount; either because of the types of message they are, or because Bob sends the same message twice (perhaps due to a noisy transmission line) and is appending a small amount of random padding (or a timestamp). Also, for simplicity, let us assume that Alice is using a public exponent of 3.

In this case Eve knows $M^3 = c_1 \pmod N$ and $(M+x)^3 = c_2 \pmod N$. She can eliminate the unknown M from these equations by using resultants, and is left with

$$x^9 + 3(c_1 - c_2)x^6 + 3(c_2^2 + 7c_1c_2 + c_1^2)x^3 + (c1 - c2)^3 \quad = \quad 0 \pmod N,$$

so she may discover the padding as long as $|x| \leq N^{1/9}$.

It is not obvious that Eve can recover M from the knowledge of x, but this is true due to a clever trick of Franklin and Reiter, which is explained and generalised in (Coppersmith *et al.*, 1996). To explain this in our case let m be a polynomial indeterminate and calculate $\gcd(m^3 - c_1, (m+x)^3 - c_2)$ over $\mathbb{Z}_N[m]$ using the Euclidean algorithm[3]. It can be shown that the result of this gcd will be the linear polynomial $m - M$ (it is clear that this divides the gcd), and hence we have discovered the original m.

The way to prevent this attack is to use more random padding. Conversely this implies that keeping the same amount of padding and increasing ones modulus N for "extra security" is a very dangerous thing to do!

Another precaution one might also choose is to distribute the random padding throughout the message, rather than in any one block. As is discussed in the next section it is

[3] Although $\mathbb{Z}_N[m]$ is not a Euclidean ring, it can be shown that if the Euclidean algorithm ever breaks, it gives a non-trivial divisor of N.

by no means certain that this action alone would prevent these kinds of attack.

4.8.4 Many small blocks of unknown plaintext

Using the theory in Section 4.7 we may sometimes uncover small blocks of plaintext even when they are placed in many locations throughout the message. See Section 4.7.4 for an example of this attack, and Theorem 4.7.2 for a description of the weak locations to place multiple blocks.

Chapter 5

Factoring

In this chapter we examine the use of lattices in factoring. The work is based very largely on a result of Coppersmith on finding small solutions of general bivariate integer equations, first shown (Coppersmith, 1996a). This result is briefly discussed in Section 5.1 in connection with the problem of factoring an integer N, when some information is known about the bits of a factor.

In Section 5.2 we show how to produce an alternative, simpler, lattice that also implies the factoring result of Coppersmith. From the work in Section 3.5 we also show (in Section 5.3) that this factoring lattice allows one to factor over the Gaussian integers and thereby find solutions to the integer equation $x^2 + y^2 = n$ for small y. This has implications on a cryptosystem proposed in (Vanstone & Zuccherato, 1997).

In Section 5.4 we modify the lattice slightly to allow for factoring integers with repeated factors, and thereby produce and analyse a factoring algorithm with an interesting new complexity.

In Section 5.5 we then modify the basic lattice in a slightly different way to search for divisors in residue classes. This leads to a constructive method to find all divisors $sx + r$ which divide N for known r, s when $s > n^{1/4}$, and an analysis to bound the number of such divisors. This problem was first considered in (Lenstra, 1984), where it was shown how to construct the divisors whenever $s > n^{1/3}$.

5.1 Coppersmith's approach

Coppersmith extended his idea of solving univariate modular equations (i.e. a univariate polynomial in x, and a linear term in y, say) to general bivariate equations in

(Coppersmith, 1996a). If we are looking for solutions to $r(x, y) = 0$ and X and Y are bounds on the sizes of $|x|$ and $|y|$ respectively, then the generalised approach works with the quantity D, which is the largest monomial of the bivariate polynomial when evaluated at $(x, y) = (X, Y)$, rather than N as in the univariate modular approach. We state Coppersmith's main theorem below.

Theorem 5.1.1 *Let $r(x, y) = \sum_{ij} r_{ij} x^i y^j$ be a bivariate polynomial over $\mathbb{Z}$ of degree δ in x and τ in y. Assume r is irreducible over $\mathbb{Z}$. Let X and Y be bounds on desired solutions x_0 and y_0. Define $D = \max_{ij} |r_{ij} X^i Y^j|$. Choose $\alpha > 0$. Assume*

$$X^{\delta + (\alpha\tau/2)} Y^{\tau + (\delta/(2\alpha))} \quad < \quad D \times 2^{-3(\delta^2 + \tau^2) - 2}.$$

In time polynomial in δ, τ and $\log_2 D$, our algorithm will produce all integer pairs $(x_0, y_0))$ with $|x_0| < X$, $|y_0| < Y$, and $r(x_0, y_0) = 0$.

Let $r(x, y)$ be as before, but with total degree δ. Assume

$$(XY)^{\delta} \quad < \quad D \times 2^{-6\delta^2 - 2}.$$

In time polynomial in δ and $\log_2 D$, our algorithm will produce all integer pairs (x_0, y_0) with $|x_0| < X$, $|y_0| < Y$, and $r(x_0, y_0) = 0$.

An interesting application of this result, and the one Coppersmith concentrated on in (Coppersmith, 1996a), is the bivariate equation $(p_0 + x)(q_0 + y) = N$. If we know the top m bits of a factor p of N, then by division we also know the top m bits of q, such that $pq = N$. This means that given p_0, and by choosing q_0 accordingly, we can ensure $D = \max\{XY, q_0 X, p_0 Y, |p_0 q_0 - N|\} \approx q_0 X \approx p_0 Y$. Let $p = N^{\alpha}$, and $X = N^{\beta}$, so $q = N^{1-\alpha}$ and $Y = N^{1-2\alpha+\beta}$. By using the first part of Coppersmith's theorem with $\delta = \tau = 1$, we must ensure that $(XY)^{3/2} < D$, i.e.

$$\left(N^{1-2\alpha+2\beta}\right)^{3/2} \quad < \quad N^{1-\alpha+\beta}$$

which will be true whenever $\beta < \alpha - 1/4$.

This implies that whenever the top $(1/4 + \varepsilon)$ bits of a factor p of N are known, the remaining bits of p may be found in polynomial time. In fact this result can be strengthened slightly, as we shall see in the next section.

5.2 An alternative method

In this section we endeavour to reach a similar factoring result to (Coppersmith, 1996a) but by following a slightly different path. In fact we follow a similar approach to that taken in Section 4.1, i.e. we produce a matrix such that all the rows correspond to polynomials that evaluate to zero to some modulus at the sought after root x_0. We then reduce this matrix using the LLL algorithm.

The situation is marginally more complicated in this case however since the modulus is *precisely* unknown, though we do have a (close) estimate to its size.

As is usual in these methods, we start by choosing an integer h which will correspond to the dimension of the matrix we form. The higher the value of h, the larger the value of permissible X (i.e. the further away our guess p_0 can be from the true divisor p), although of course, a larger matrix also implies a greater time reducing it.

We then choose an integer $u < h$ and form the matrix $M(h, u, X)$ with rows corresponding[1] to the polynomials

$$p_i(x) = \begin{cases} N^{u-i}(p_0 + x)^i & 0 \le i \le u \\ (p_0 + x)^u x^{i-u} & u < i \le h. \end{cases} \tag{5.1}$$

For a given h the optimum choice of u is given by equation 5.3, and one should choose X to be as large as possible, but still satisfy equation 5.4.

Thus, as an example, if $h = 4$ and $u = 2$ then we form the following matrix.

$$M(4, 2, X) = \begin{pmatrix} N^2 & & & \\ Np_0 & NX & & \\ p_0^2 & 2p_0X & X^2 & \\ & p_0^2X & 2p_0X^2 & X^3 \end{pmatrix}$$

Clearly all the rows of such a matrix correspond to polynomials which evaluate to zero modulo $(p_0 + x_0)^u = p^u$ at $x = x_0$, thus so does any linear combination of them.

If we LLL reduce this matrix to form a small row b_1 then for a general u, h and X we

[1]Having decided on the natural number X the row that corresponds to the polynomial $p(x) = a_n x^n + \ldots + a_2 x^2 + a_1 x + a_0$, $n < h$ is of size h and given by

$$\left[\begin{array}{ccccccc} a_0 & a_1 X & a_2 X^2 & \ldots a_n X^n & 0 & \ldots & 0 \end{array} \right].$$

have

$$\|b_1\| \quad < \quad 2^{(h-1)/4}\left(N^{u(u+1)/2}X^{h(h-1)/2}\right)^{1/h},$$

from equation 3.14.

Letting $b_1(x)$ denote the polynomial corresponding to b_1 then, in a similar way to Section 4.1, we have

$$\begin{aligned} |b_1(x)| \quad &< \quad \sqrt{h}\,\|b_1\| \\ &= \quad 2^{(h-1)/4}\sqrt{h}N^{u(u+1)/2h}X^{(h-1)/2} \qquad \text{for all } |x| < X. \end{aligned}$$

Thus, if X was such that $|b_1(x)| < (p_0 + x_0)^u \simeq p_0^u$ for all $|x| < X$, when we already know $b_1(x_0) = 0 \pmod{(p_0 + x_0)^u}$ this would imply that $b_1(x_0) = 0$, i.e. we only need to solve $b_1(x)$ over the integers.

Letting $p_0 = N^{\alpha}$, this means we want to find the maximum X such that

$$2^{(h-1)/4}\sqrt{h}N^{u(u+1)/2h}X^{(h-1)/2} \quad < \quad N^{u\alpha},$$

which turns out to be

$$X \quad < \quad \left(2^{-1/2}h^{-1/(h-1)}\right)N^{u(2\alpha h-u-1)/(h(h-1))}. \tag{5.2}$$

For a given h and $\alpha \simeq \log_N(p_0)$ we can maximise the r.h.s. by choosing

$$u \quad = \quad \alpha h - \frac{1}{2}, \tag{5.3}$$

and so the maximum allowable X (i.e. the maximum allowable error $|p - p_0|$) must satisfy

$$X \quad < \quad \left(2^{-1/2}h^{-1/(h-1)}\right)N^{(\alpha h-1/2)^2/(h(h-1))}. \tag{5.4}$$

Observe that as $h \to \infty$ the maximum allowable $X \to N^{(\alpha^2)}$, which we now state this result precisely, as a general theorem.

Theorem 5.2.1 *All $x \in \mathbb{Z}$ such that $(p_0 + x)$ divides N where $p_0 = N^{\alpha}$ and $|x| < N^{\gamma}$ can be found in polynomial time whenever there exist integers $h > u > 0$ such that*

$$\gamma h(h-1) - 2u\alpha h + u(u+1) \quad < \quad 0.$$

The largest value of γ for which this can hold is $\alpha^2 - \varepsilon$.

Proof: The proof follows from equation 5.2 with $X = N^\gamma$. □

Note that with $\alpha = 1/2$ we have Coppersmith's result that only the top $(1/4+\varepsilon)\log_2 N$ bits of p need to be known to find it exactly, but for smaller α this result becomes superior. However, although Coppersmith does not explicitly give the bounds as in Theorem 5.2.1, he does explain how his algorithm can be modified to achieve such improved bounds.

Also notice that this approach seems to eliminate the need for the resultant calculations used in (Coppersmith, 1996a), because it solves with respect to only one variable. However, given two equations in x and y, with one equation being $xy = N$, then one could replace y by N/x in the other and consider the numerator (a polynomial in x) of the subsequent expression. This shows that Coppersmith's use of resultants was trivial in this case.

5.3 Factoring over the Gaussian integers

Essentially there is nothing to do to extend the factoring method to Gaussian integers, but it should be shown that all the necessary arguments still hold.

Firstly notice that the LLL algorithm was shown to extend to unitary lattices in Section 3.5, and that this ensures that a lattice vector b_1 is found such that

$$||b_1|| \quad \leq \quad 2^{(n-1)/2}\Delta^{1/n}.$$

Now following the exposition of Section 5.2, but letting the symbol Z replace X to illustrate that this is now the size of a complex number, one creates the same matrix $M(h, u, Z)$ but with N and p_0 as Gaussian integers, and hence (using equation 3.28) we can find a Gaussian integer polynomial $b_1(z)$ such that for all $|z| < Z$ we have

$$|b_1(z)| \quad < \quad 2^{(h-1)/2}\sqrt{h}\,|N|^{u(u+1)/2h} Z^{(h-1)/2}.$$

If this is less than $|p|^u$ then $b_1(z)$ will be true over the Gaussian integers, and can therefore be solved in polynomial time (for instance by substituting $z = x + iy$ and then using the resultant algorithm on the real and complex parts).

By balancing u as in Section 5.2 we achieve a similar result to Theorem 5.2.1 which we state precisely below, again using $\mathcal{G}$ to denote the Gaussian integers.

Theorem 5.3.1 *All $z \in \mathcal{G}$ such that $(p_0 + z)$ divides $N \in \mathcal{G}$ where $p_0 \in \mathcal{G}$, $|p_0| = |N|^\alpha$ and $|z| < |N|^\gamma$ can be found in polynomial time whenever there exist integers $h > u > 0$ such that*

$$\gamma h(h-1) - 2u\alpha h + u(u+1) \quad < \quad 0.$$

The largest value of γ for which this can hold is $\alpha^2 - \varepsilon$.

5.3.1 An application

Given the Diophantine equation

$$x^2 - y^2 = N, \quad \text{where } y = kN^{1/4}, \tag{5.5}$$

it can easily be shown that x is bounded by

$$\sqrt{N} \quad \leq \quad x \quad \leq \quad \sqrt{N} + O(k^2),$$

thus integer solutions may be found to equation 5.5 after $O(k^2)$ tests of x. We note that a similar approach can be used to find solutions to

$$x^2 + y^2 = N, \quad \text{where } y = kN^{1/4}. \tag{5.6}$$

Fermat applied this method to factorise the number

$$N = pq = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2,$$

and pointed out that if $(p - q) \leq kN^{1/4}$ then this gives rise to an $O(k^2)$ factorisation method. It is a clear warning to cryptographers to ensure that the primes used in the RSA protocol for instance are not close enough together for this attack to be feasible.

The cryptographic situation is actually seen to be weaker that this, with the aid of Coppersmith's result, since equation 5.5 factors as

$$\begin{aligned} N &= (x-y)(x+y) \\ &= \left(\sqrt{N} + O(k^2) - kN^{1/4}\right)\left(\sqrt{N} + O(k^2) + kN^{1/4}\right) \\ &= \left(\sqrt{N} - k\left(N^{1/4} - O(k)\right)\right)\left(\sqrt{N} + k\left(N^{1/4} + O(k)\right)\right). \end{aligned}$$

Assuming that $k < N^{1/4}$ this means that there is a Gaussian integer factor p of N at about $p \simeq \sqrt{N} - ikN^{1/4}$, and Theorem 5.2.1 implies that for any $p_0 \simeq N^{1/2}$ we can find the true p if $p = p_0 + x_0$ with $|x_0| < N^{1/4-\epsilon}$. Together these facts imply that p can be found after $O(k^{1+\epsilon})$ trials of possible p_0.

Using the Gaussian integer methods outline above we can now use an identical approach to solve equation 5.6 since it factors as

$$\begin{aligned} N &= (x-iy)(x+iy) \\ &= \left(\sqrt{N} - O(k^2) - ikN^{1/4}\right)\left(\sqrt{N} - O(k^2) + ikN^{1/4}\right) \\ &= \left(\sqrt{N} - k\left(iN^{1/4} + O(k)\right)\right)\left(\sqrt{N} + k\left(iN^{1/4} - O(k)\right)\right). \end{aligned}$$

Assuming that $k < N^{1/4}$ this means that there is a factor p of N at about $p \simeq \sqrt{N} - kN^{1/4}$, and Theorem 5.3.1 implies that for any $p_0 \simeq N^{1/2}$ we can find the true p if $p = p_0 + z_0$ with $|z_0| < N^{1/4-\epsilon}$. Together these facts imply that p can be found after $O(k^{1+\epsilon})$ trials of possible p_0.

It was shown in (McKee & Pinch, 1998) and (Coppersmith, 1998) that the cryptosystem in (Vanstone & Zuccherato, 1997) was susceptible to attack from the fact that it produces a number $N = pq$ which is the product of two primes of the form either $a^2 + 4$ or $a^2 - 3a + 9$.

For simplicity let us restrict ourselves to the case $p = a^2 + 4$ and $q = b^2 + 4$, thus $N = pq = (ab+4)^2 + (2(a+b))^2$ (one can either verify this algebraically or with the matrix equations below).

$$\begin{aligned} \det\begin{pmatrix} a & 2 \\ -2 & a \end{pmatrix} \det\begin{pmatrix} b & 2 \\ -2 & b \end{pmatrix} &= \det\left[\begin{pmatrix} a & 2 \\ -2 & ab \end{pmatrix}\begin{pmatrix} b & 2 \\ -2 & b \end{pmatrix}\right] \\ &= \det\begin{pmatrix} ab+4 & 2(a+b) \\ -2(a+b) & ab+4 \end{pmatrix} \end{aligned}$$

Let $x = ab + 4$ and $y = 2(a+b)$, so $N = x^2 + y^2$, and let k be such that $y = kN^{1/4}$ then assuming $a > b$, we have that $k = O(a/b)$. Both McKee and Pinch, and Coppersmith realised this weakness and quoted the $O((a/b)^2)$ "Fermat method" to break the cryptosystem. However using the Gaussian integer variant of the factoring algorithm of Section 5.2 this can be improved to an $O(a/b)$ attack, approximately square-rooting[2] the time necessary to break Vanstone and Zuccherato's cryptosystem.

5.4 Factoring numbers with repeated factors

In this section we show there is a deterministic $O(p^{1-m\alpha})$ time algorithm for factoring integers of the form $N = p^m q$ where $p = N^\alpha$. This has implications in a few recently proposed cryptosystems (see e.g. (Okamoto & Uchiyama, 1998) and (Takagi, 1998)) which use moduli of the form $N = p^2 q$, showing that there is at worst an $O(N^{1/8})$ algorithm for factoring such N, and that the method can be improved to $O(N^{1/9})$ if p and q are chosen to be of the same magnitude (i.e. $N^{1/3}$).

5.4.1 The method

Following the exposition given in Section 5.2, let us firstly assume that we have an approximation p_0 to a true factor $p = p_0 + x_0$ of N, where $N = p^m q$. We shall show that if $|x_0|$ is sufficiently small then p may be found in polynomial time.

Theorem 5.4.1 *All $x \in \mathbb{Z}$ such that $(p_0 + x)^m$ divides N where $p_0 = N^\alpha$ and $|x| < N^\gamma$ can be found in polynomial time whenever there exist integers $h > u > 0$ such that*

$$\gamma h(hm-1) - 2u\alpha hm + u(u+1) \quad < \quad 0.$$

The largest value of γ for which this can hold is $m\alpha^2 - \varepsilon$.

Proof: The following is a relatively minor extension to the proof of Theorem 5.2.1, but is shown in detail for completeness sake. For given integers h and $u < h$ (which are specified in more detail later) consider the polynomials

$$p_{i,j}(x) = x^j N^{u-i}(p_0 + x)^{im} \quad \begin{cases} (0 \le i < u, \quad 0 \le j < m), \text{ and} \\ (i = u, \quad 0 \le j \le (h-u-1)m). \end{cases}$$

For all $0 \le i < h$ we have that $p_i(x_0) = 0 \pmod{p^{um}}$, so any linear combination of these polynomial must also evaluate to zero modulo p^{um} at $x = x_0$.

Let X be an upper bound on the size of $|x_0|$ (the maximum possible value for X is also calculated later). We firstly form a $(hm) \times (hm)$ matrix $M_{h,u} = (m_{i,j})$ where the

[2] Having said this the original parameters suggested by Vanstone and Zuccherato had $a \approx b$ in which case the cryptosystem was flawed even by an $O((a/b)^2)$ attack.

entry $m_{i,j}$ is the coefficient of x^j in $p_{a,b}(x)$ multiplied by X^j, and where a and b are such that $b + am = i$, with $a \leq u$, and $0 \leq b < m$ for all $a < u$. For instance, if $m = 2$, and with $h = 4$ and $u = 2$ we would consider the matrix

$$M_{4,2} = \begin{pmatrix} N^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & N^2X & 0 & 0 & 0 & 0 & 0 & 0 \\ p_0^2N & 2p_0NX & NX^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & p_0^2NX & 2p_0NX^2 & NX^3 & 0 & 0 & 0 & 0 \\ p_0^4 & 4p_0^3X & 6p_0^2X^2 & 4p_0X^3 & X^4 & 0 & 0 & 0 \\ 0 & p_0^4X & 4p_0^3X^2 & 6p_0^2X^3 & 4p_0X^4 & X^5 & 0 & 0 \\ 0 & 0 & p_0^4X^2 & 4p_0^3X^3 & 6p_0^2X^4 & 4p_0X^5 & X^6 & 0 \\ 0 & 0 & 0 & p_0^4X^3 & 4p_0^3X^4 & 6p_0^2X^5 & 4p_0X^6 & X^7 \end{pmatrix}.$$

We then LLL reduce the rows of $M_{h,u}$ to find a small row b_1 which satisfies

$$\begin{aligned} |b_1| &< 2^{(h-1)/4}(\det M_{h,u})^{1/(hm)} \\ &< 2^{(h-1)/4}\left(N^{mu(u+1)/2}X^{hm(hm-1)/2}\right)^{1/(hm)}. \end{aligned}$$

Letting $b_1(x)$ be the polynomial corresponding to this small row, then we have that for all $|x| < X$

$$\begin{aligned} |b_1(x)| &< \sqrt{(hm)}\,|b_1| \\ &< 2^{(h-1)/4}\sqrt{(hm)}\,N^{u(u+1)/(2h)}X^{(hm-1)/2}. \end{aligned}$$

Note that if

$$|b_1(x)| < p^{um}, \tag{5.7}$$

then $b_1(x_0) = 0$ over the integers, since we know that $b_1(x_0) = 0 \pmod{p^{um}}$.

For a given h and $p_0 \simeq p = N^\alpha$ we now calculate the optimum choice of u. To ensure that condition 5.7 is satisfied (and thus we can find x_0 by finding the linear factors of b_1 over the integers) we must have (ignoring the small factors independent of N) that

$$N^{u(u+1)/(2h)}X^{(hm-1)/2} < N^{\alpha um}$$

which will occur whenever

$$X < N^{u(2\alpha hm-u-1)/(h(hm-1))}.$$

The right-hand side of the above equation is maximised by choosing $u = hm\alpha - 1/2$ in which case we can choose $X = N^{\gamma}$ where

$$\gamma = \frac{(2\alpha hm - 1)^2}{4h(hm-1)}.$$

Notice that as $h \to \infty$ we have that $X \to N^{m\alpha^2}$ which completes the proof. $\square$

Note that if one were to choose an α to minimise the risk from this attack, i.e. $\alpha = 1/m$ (which makes the attack an $O(\sqrt{p})$ method) then $p = N^{\alpha}$ may well be small enough for the elliptic curve factorisation method (ECM) to work. Alternatively, if one chooses p and q approximately the same size, i.e. $\alpha = 1/(m+1)$, then one has an $O(p^{1/(m+1)})$ method of factoring.

This practical running speed of this algorithm (in comparison with ECM) has been studied in more detail in (Boneh *et al.*, 1999).

5.5 Divisors in residue classes

Let r, s, n be integers satisfying $0 \leq r < s < N$, $s \geq N^{\alpha}$, $\alpha > 1/4$, and $\gcd(r, s) = 1$. Lenstra showed in (Lenstra, 1984) that the number of integer divisors of N equivalent to $r \pmod s$ satisfies an upper bound $c(\alpha)$ dependent only on α. He then proved the bound $c(\alpha) = O((\alpha - 1/4)^{-2})$, and showed how to construct all such divisors when $\alpha > 1/3$, in time polynomial in $\log N$ and $(\alpha - 1/4)^{-1}$. By comparison we show how to construct all such divisors when $\alpha > 1/4$ in polynomial time, and also improve the asymptotic analysis of the bound to $O((\alpha - 1/4)^{-3/2})$. However, at present, the actual bounds achieved from these techniques for $\alpha > 1/3$ are inferior to Lenstra's results (see Section 5.5.4) and in fact the bound of $c(\alpha) = O((\alpha - 1/4)^{-3/2})$ can be reached from a more careful analysis of Lenstra's technique, as shown in (Coppersmith *et al.*, 1998).

The fact that one can construct the divisors in residue classes follows fairly trivially from (Coppersmith, 1996a), as is shown in Section 5.5.1. However in Section 5.5.2 we extend Theorem 5.2.1 because of its simpler lattice, and for a simpler analysis in calculating bounds on $c(\alpha)$. These bounds are shown in Section 5.5.3, and a further discussion on the subject is given in Section 5.5.4.

The author is very grateful for much helpful cooperation with Don Coppersmith and S. V. Nagaraj in the work of this section.

5.5.1 An application to RSA

In this section we describe an attack given in (Boneh *et al.*, 1998) which makes use of divisors in residue classes. We firstly show how to use Theorem 5.1.1 to calculate the divisors of a number N which lie in given residue classes.

Theorem 5.5.1 *All divisors of N of the form $sx+r$, for some known s and r, where $0 \leq r < s < N$, $\gcd(r,s) = 1$ and $s = N^{\alpha}$ may be found in time polynomial in $\log N$ and $(\alpha - 1/4)^{-1}$ whenever $\alpha > 1/4$.*

Proof: Suppose (sx_0+r) divides N for some known s and r, i.e. $N = (sx_0+r)(sy_0+r')$ where $r' = N/r \pmod{s}$ (note r' exists otherwise we can find a factor of N). Then the polynomial

$$\begin{aligned} p(x,y) &= \frac{1}{s}\left((sx+r)(sy+r') - N\right) \\ &= sxy + r'x + ry + (rr' - N)/s \end{aligned}$$

will be zero when evaluated at $(x,y) = (x_0, y_0)$.

If $s = N^{\alpha}$, and $|x_0| < X = N^{\beta}$ then $|y_0| < Y = N^{1-2\alpha-\beta}$ and $D = sXY = N^{1-\alpha}$. To use Theorem 5.1.1 we must ensure that $(XY)^{3/2} < D$, which will be true whenever $\alpha > 1/4$. □

Now suppose we are using RSA with $N = pq$, where $p \approx q \approx \sqrt{N}$, and some small public exponent e. Also assume (for simplicity) that $\gcd(p-1, q-1) = 2$, then by the defining property of the encryption and decryption exponents e and d we have

$$ed + k\left(\frac{N+1}{2} - \frac{p+q}{2}\right) = 1.$$

Now suppose that one has exposed the bottom 1/4 of the bits of d perhaps by the timing techniques as described in (Kocher, 1996) and Section 1.2.5. Then, assuming N is an n-bit integer, we have

$$ed + k(M - t) = 1 \pmod{2^{n/4}},$$

where $M = (N+1)/2$ and only k and $t = (p+q)/2$ are unknown modulo $2^{n/4}$. Since we are assuming a low e, and we know $k/e \approx d/N \approx 1$, we can iterate through k giving possible choices for t. For each of these candidate values for $t \pmod{2^{n/4}}$ we can then work out the implied possible divisor $p \pmod{2^{n/4}}$ of N by solving the following quadratic equation

$$p^2 - tp + N = 0 \pmod{2^{n/4}}.$$

And finally, with the knowledge of $p \pmod{2^{n/4}}$ we can then calculate the remaining bits of p by using Theorem 5.5.1.

There are various other attacks on RSA described in (Boneh *et al.*, 1998), which make use of divisors in residue classes, which the interested reader may consult.

5.5.2 The method

In this section we extend Theorem 5.2.1 to account for divisors of N of the form $(sx+r)$ for some known s and r. We then form a bound on the number of such divisors by considering the degree of the polynomial derived from the lattice methods.

To extend Theorem 5.2.1 to divisors in residue classes we firstly assume that the size of x is within known bounds.

Lemma 5.5.2 *All x such that $(sx+r)$ divides N where $0 \le r < s < N$, $\gcd(r,s) = 1$, $s = N^\alpha$ and $N^\beta \le |x| < N^\gamma$ can be found in polynomial time whenever there exist integers $h > u > 0$ such that*

$$\gamma h(h-1) - 2u(\alpha+\beta)h + u(u+1) < 0. \tag{5.8}$$

The largest value of γ for which this can hold is $(\alpha+\beta)^2 - \varepsilon$.

Proof: Let s' be such that $ss' = 1 \pmod{N}$, then the following are divisible by $(sx+r)$:

$$\begin{aligned} & N, \quad \text{and} \\ s'(sx+r) &= x + r'' \pmod{N}. \end{aligned}$$

Thus we form the matrix exactly as in Section 5.2 with $p_0 = r''$. Now all the rows are equivalent to multiples of $(sx+r)^u$ which is at least $N^{(\alpha+\beta)u}$ under the assumption that $|x| \ge N^\beta$. Following the analysis in Section 5.2 the only change to equation 5.2 is that α becomes $\alpha+\beta$ which yields the required result.

The best choice of u is $(\alpha+\beta)h - 1/2$ implying $\lim_{h\to\infty} \gamma_{\max} = (\alpha+\beta)^2$. □

We now show that when $\alpha > 0.365$ we can use Lemma 5.5.2 directly to find all the relevant divisors.

Corollary 5.5.3 *All divisors of N of the form $(sx+r)$ with $s = N^\alpha$ and $(\sqrt{3}-1)/2 < \alpha \leq 1/2$ may be found in polynomial time.*

Proof: We first find the divisors $(sx+r) < \sqrt{N}$ and then the divisors $(sx+r) > \sqrt{N}$ are found by looking for their corresponding divisors $(sx+r') < \sqrt{N}$ where $r' = N/r \pmod{s}$. Since $sx < \sqrt{N}$ we know that $x < N^{1/2-\alpha}$, but x could possibly be as small as 1, so we set $\beta = 0$ and $\gamma = 1/2 - \alpha$ and apply Lemma 5.5.2. We have that $(1/2-\alpha) < \alpha^2$ whenever $\alpha > (\sqrt{3}-1)/2 \approx 0.365$. □

When $\alpha < (\sqrt{3}-1)/2$ we cannot find all the divisors with one choice of $X = N^{1/2-\alpha}$; instead we must split the interval $[0, 1/2-\alpha]$ up in to more intervals and apply Lemma 5.5.2 to each of these in turn.

Corollary 5.5.4 *By reducing two matrices, we can find all divisors of N of the form $(sx+r)$ where $s = N^\alpha$ and $\alpha > 0.32066$ (approximately).*

Proof: We split the interval $[0, 1/2-\alpha]$ in to $[0, \delta]$ and $[\delta, 1/2-\alpha]$. Applying Lemma 5.5.2, all the solutions in the first interval will be found as long as $\delta < \alpha^2$ and all the solutions in the second interval will be found if $(1/2-\alpha) < (\alpha+\delta)^2$. Thus if we let $\alpha_0 \approx 0.32066$ be the solution to $(\alpha+\alpha^2)^2 + \alpha - 1/2 = 0$ and $\delta = \alpha_0^2$ then both of these will hold. □

Lemma 5.5.5 *All divisors of N of the form $(sx+r)$ with $s = N^\alpha$ and $\alpha > 1/4$ may be found in polynomial time.*

Proof: By splitting the interval $[0, 1/2-\alpha]$ in to

$$[0, \delta_1], [\delta_1, \delta_2], \ldots [\delta_J, 1/2-\alpha],$$

we require that

$$\begin{aligned}
1/2 - \alpha &< (\alpha + \delta_J)^2 \\
\delta_J &< (\alpha + \delta_{J-1})^2 \\
&\vdots \\
\delta_1 &< \alpha^2
\end{aligned}$$

This will be true for all $\alpha > \alpha_0$ where α_0 is a solution to

$$\begin{aligned} f_J(\alpha) &= \left(\alpha + \left(\alpha + \ldots \left(\alpha + \alpha^2\right)^2\right)^2 \ldots\right)^2 + \alpha - 1/2 \\ &= 0 \end{aligned}$$

Notice that we have $f_J(\alpha) = (f_{J-1}(\alpha) + 1/2)^2 + \alpha - 1/2$ which implies that $f_J(1/4) \to 0$ as $J \to \infty$, i.e. $\alpha_0 \to 1/4$ as $J \to \infty$. □

Rather than split the intervals to minimise the number of different matrices that need to be reduced (i.e. different values of X with very large h) as in Lemma 5.5.5, we now aim to split the intervals so as to minimise the density of the possible solutions in each interval[3], i.e. $(h-1)/(\gamma - \beta)$. This leads to the following result.

Lemma 5.5.6 *For any given s, r such that $o \leq r < s < N$, $\gcd(r, s) = 1$ and $s = N^\alpha$ the number of divisors of N of the form $(sx + r)$ is upper bounded by*

$$c(\alpha) = 2 + \frac{\pi\alpha}{(\alpha - 1/4)^{3/2}} + \frac{4\alpha}{\alpha - 1/4}.$$

Proof: Assume we are given $0 < \alpha < 1$. As before we consider the divisors $(sx + r) < \sqrt{N}$ first. For any $0 \leq \beta \leq 1/2 - \alpha$ where $|x| \geq N^\beta$ we can choose h and u to imply a γ (from equation 5.8), denoted $\Gamma(\beta)$, which minimises $(h-1)/(\gamma - \beta)$. These values are

$$\begin{aligned} h &= \left\lceil \frac{2\alpha}{(\alpha+\beta)^2 - \beta} \right\rceil, \\ u &= \lfloor (\alpha + \beta) h \rfloor, \\ \Gamma(\beta) &= \frac{2(\alpha+\beta)uh - u(u+1)}{h(h-1)} + \varepsilon. \end{aligned}$$

The density then satisfies

$$\frac{h-1}{\Gamma(\beta) - \beta} < \frac{4\alpha}{((\alpha+\beta)^2 - \beta)^2}.$$

Since the r.h.s. is an increasing function for all $\beta < 1/2 - \alpha$ we have that

$$h - 1 < (\Gamma(\beta) - \beta)\frac{4\alpha}{((\alpha+\beta)^2 - \beta)^2} < \int_{v=\beta}^{\Gamma(\beta)} \frac{4\alpha}{((\alpha+v)^2 - v)^2} dv.$$

[3] A third (unexplored) option would be to split the intervals so as to minimise the *expected time* to find all the relevant divisors.

If we split the interval $[0, 1/2 - \alpha]$ in to

$$[0, \Gamma(0)),\ [\Gamma(0), \Gamma(\Gamma(0))),\ \ldots,\ [\Gamma^{(i-1)}(0),\ \Gamma^{(i)}(0))$$

where $\Gamma^{(i)}(0) > 1/2 - \alpha$ and then sum $(h-1)$ (the bound on the possible number of divisors in each interval) over all these intervals we have.

$$\begin{aligned}\sum(h-1) &< \left(\int_0^{1/2-\alpha} \frac{4\alpha}{((\alpha+v)^2 - v)^2} dv\right) + \frac{2\alpha}{\alpha - 1/4} \\ &< \frac{\pi\alpha}{(\alpha - 1/4)^{3/2}} + \frac{2\alpha}{\alpha - 1/4}\end{aligned}$$

We must also account for the fact that we may have $r = 1$ in which case $(sx + r)$ will divide N when $x = 0$, and so we should add one to this total. The same bound applies to the divisors $(sx + r) > \sqrt{N}$ by considering their corresponding divisors as before, and so we reach the desired formula above. $\square$

5.5.3 Results

In this section use Lemma 5.5.2 to calculate the bounds on the number of divisors in residue classes for particular α.

Example 5.1 *For instance using just two intervals, $J = 2$, and having $h_1 = 3$, $u_1 = 1$ for the first interval, and also $h_2 = 3$, $u_2 = 1$ for the second interval, then to find the relevant divisors $(sx + r)$ we would reduce the following four lattices, where $r' = N/r \pmod{s}$ and $p_1 = r/s \pmod{N}$, $p_2 = r'/s \pmod{N}$, $X_1 = N^{1/2-\alpha}$ and $X_2 = N^{5/6-2\alpha}$.*

β	γ	$(sx+r) \leq \sqrt{N}$	$(sx+r) > \sqrt{N}$
$\frac{5}{6} - 2\alpha$	$\frac{1}{2} - \alpha$	$\begin{pmatrix} 0 & 0 & N \\ 0 & X_1 & p_1 \\ X_1^2 & p_1 X_1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & N \\ 0 & X_1 & p_2 \\ X_1^2 & p_2 X_1 & 0 \end{pmatrix}$
$\frac{7}{6} - 3\alpha$	$\frac{5}{6} - 2\alpha$	$\begin{pmatrix} 0 & 0 & N \\ 0 & X_2 & p_1 \\ X_2^2 & p_1 X_2 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & N \\ 0 & X_2 & p_2 \\ X_2^2 & p_2 X_2 & 0 \end{pmatrix}$

Thus if $7/6 - 3\alpha < 0$, i.e. $\alpha > 7/18 \approx 0.389$ reducing these lattices will discover all the divisors of N of the form $(sx + r)$ except the two possible divisors of 1 and N itself. Thus there are at most $4 \times (3 - 1) + 2 = 10$ divisors of N when $\alpha > 7/18$.

However since the lattice methods find the divisors $(sx+r)$ where $|x|$ is small, we also find the divisors of N of the form $(s(-x)+r) = -(sx-r)$. Thus this result can be strengthened to there being at most 10 divisors of N either of the form $sx+r$ or $sx-r$.

For a given number of intervals we can choose the h_i and u_i so as to minimise the α for which the divisors will be found for a given $H = \sum h_i$ (the bound $c(\alpha)$ is only dependent on H). For instance this has been done for the case of two intervals below.

$\alpha >$	$c(\alpha) \leq$	h_1	h_2	u_1	u_2
$7/18 \approx 0.389$	10	3	3	1	1
$3/8 = 0.375$	12	3	4	1	1
$7/19 \approx 0.368$	14	4	4	1	1
$29/80 \approx 0.363$	16	5	4	2	1
$5/14 \approx 0.357$	18	6	4	2	1
$39/110 \approx 0.355$	20	6	5	2	1
$43/122 \approx 0.353$	22	6	6	2	2
$4339/13039 \approx 0.333$	58	17	13	7	4

The following table, which ranges over the next three pages, shows the optimum number of intervals, J, and the corresponding h_i, u_i, to minimise α for $6 \leq c(\alpha) \leq 132$. This covers all $\alpha > 0.29$.

J	H	$c(\alpha) \leq$	$\alpha >$ (approx.)	$\alpha >$ (exact)	h_i	u_i
1	3	6	0.416667	$5/12$	3	1
1	4	8	0.400000	$2/5$	4	1
2	6	10	0.388889	$7/18$	$3,3$	$1,1$
2	7	12	0.375000	$3/8$	$3,4$	$1,1$
2	8	14	0.368421	$7/19$	$4,4$	$1,1$
2	9	16	0.362500	$29/80$	$5,4$	$2,1$
2	10	18	0.357143	$5/14$	$6,4$	$2,1$
3	12	20	0.353846	$23/65$	$4,4,4$	$1,1,1$
3	13	22	0.346429	$97/280$	$5,4,4$	$2,1,1$
3	14	24	0.342975	$83/242$	$6,4,4$	$2,1,1$
3	15	26	0.340000	$17/50$	$5,6,4$	$2,2,1$
3	16	28	0.337209	$29/86$	$6,6,4$	$2,2,1$
3	17	30	0.334783	$77/230$	$6,6,5$	$2,2,1$

J	H	$c(\alpha) \leq$	$\alpha >$ (approx.)	$\alpha >$ (exact)	h_i	u_i
4	19	32	0.331325	55/166	5, 6, 4, 4	2, 2, 1, 1
4	20	34	0.329337	467/1418	6, 6, 4, 4	2, 2, 1, 1
4	21	36	0.327453	761/2324	7, 6, 4, 4	3, 2, 1, 1
4	22	38	0.325532	153/470	5, 6, 6, 5	2, 2, 2, 1
4	23	40	0.323892	263/812	7, 6, 6, 4	3, 2, 2, 1
4	24	42	0.321733	2117/6580	7, 6, 6, 5	3, 2, 2, 1
4	25	44	0.320204	943/2945	8, 6, 6, 5	3, 2, 2, 1
4	26	46	0.318837	1261/3955	8, 6, 7, 5	3, 2, 2, 1
5	28	48	0.317414	6799/21420	7, 6, 6, 4, 5	3, 2, 2, 1, 1
5	29	50	0.316170	1007/3185	8, 6, 6, 4, 5	3, 2, 2, 1, 1
5	30	52	0.314856	11593/36820	7, 6, 6, 6, 5	3, 2, 2, 2, 1
5	31	54	0.313679	6719/21420	7, 6, 6, 7, 5	3, 2, 2, 2, 1
5	32	56	0.312304	1193/3820	7, 8, 6, 6, 5	3, 3, 2, 2, 1
5	33	58	0.311004	1611/5180	7, 8, 6, 7, 5	3, 3, 2, 2, 1
5	34	60	0.310045	10016/32305	8, 8, 6, 7, 5	3, 3, 2, 2, 1
5	35	62	0.309127	779/2520	9, 8, 6, 7, 5	4, 3, 2, 2, 1
5	36	64	0.308227	26301/85330	10, 8, 6, 7, 5	4, 3, 2, 2, 1
5	37	66	0.307330	41047/133560	9, 8, 8, 7, 5	4, 3, 3, 2, 1
5	38	68	0.306459	5599/18270	9, 8, 9, 7, 5	4, 3, 3, 2, 1
5	39	70	0.305619	5113/16730	10, 8, 9, 7, 5	4, 3, 3, 2, 1
6	41	72	0.304941	6233/20440	7, 8, 8, 6, 7, 5	3, 3, 3, 2, 2, 1
6	42	74	0.304292	77246/253855	8, 8, 8, 6, 7, 5	3, 3, 3, 2, 2, 1
6	43	76	0.303277	223163/735840	9, 8, 8, 6, 7, 5	4, 3, 3, 2, 2, 1
6	44	78	0.302601	42703/141120	9, 8, 8, 7, 7, 5	4, 3, 3, 2, 2, 1
6	45	80	0.301940	38911/128870	10, 8, 8, 7, 7, 5	4, 3, 3, 2, 2, 1
6	46	82	0.301166	43639/144900	9, 8, 8, 9, 7, 5	4, 3, 3, 3, 2, 1
6	47	84	0.300560	39743/132230	10, 8, 8, 9, 7, 5	4, 3, 3, 3, 2, 1
6	48	86	0.299811	3169/10570	9, 10, 8, 9, 7, 5	4, 4, 3, 3, 2, 1
6	49	88	0.299251	51929/173530	10, 10, 8, 9, 7, 5	4, 4, 3, 3, 2, 1
6	50	90	0.298757	69473/232540	11, 10, 8, 9, 7, 5	5, 4, 3, 3, 2, 1
6	51	92	0.298225	6436/21581	12, 10, 8, 9, 7, 5	5, 4, 3, 3, 2, 1
6	52	94	0.297753	270997/910140	11, 10, 10, 9, 7, 5	5, 4, 4, 3, 2, 1
6	53	96	0.297178	144161/485100	11, 10, 11, 9, 7, 5	5, 4, 4, 3, 2, 1
6	54	98	0.296665	146996/495495	12, 10, 11, 9, 7, 5	5, 4, 4, 3, 2, 1

J	H	$c(\alpha) \leq$	$\alpha >$ (approx.)	$\alpha >$ (exact)	h_i	u_i
7	56	100	0.296248	24159/81550	$9, 10, 8, 8, 9, 7, 5$	$4, 4, 3, 3, 3, 2, 1$
7	57	102	0.295679	4243/14350	$9, 10, 10, 9, 7, 7, 5$	$4, 4, 4, 3, 2, 2, 1$
7	58	104	0.295248	69463/235270	$10, 10, 10, 9, 7, 7, 5$	$4, 4, 4, 3, 2, 2, 1$
7	59	106	0.294682	93031/315700	$11, 10, 10, 9, 7, 7, 5$	$5, 4, 4, 3, 2, 2, 1$
7	60	108	0.294256	8612/29267	$12, 10, 10, 9, 7, 7, 5$	$5, 4, 4, 3, 2, 2, 1$
7	61	110	0.293781	283669/965580	$11, 10, 10, 9, 9, 7, 5$	$5, 4, 4, 3, 3, 2, 1$
7	62	112	0.293381	26252/89481	$12, 10, 10, 9, 9, 7, 5$	$5, 4, 4, 3, 3, 2, 1$
7	63	114	0.292973	20303/69300	$11, 10, 10, 11, 9, 7, 5$	$5, 4, 4, 4, 3, 2, 1$
7	64	116	0.292595	1467028/5013855	$12, 10, 10, 11, 9, 7, 5$	$5, 4, 4, 4, 3, 2, 1$
7	65	118	0.292177	178181/609840	$11, 12, 10, 11, 9, 7, 5$	$5, 5, 4, 4, 3, 2, 1$
7	66	120	0.291819	362599/1242549	$12, 12, 10, 11, 9, 7, 5$	$5, 5, 4, 4, 3, 2, 1$
7	67	122	0.291498	357583/1226709	$12, 12, 10, 11, 9, 8, 5$	$5, 5, 4, 4, 3, 2, 1$
7	68	124	0.291178	1442773/4954950	$14, 12, 10, 11, 9, 7, 5$	$6, 5, 4, 4, 3, 2, 1$
7	69	126	0.290836	764769/2629550	$14, 12, 11, 11, 9, 7, 5$	$6, 5, 4, 4, 3, 2, 1$
7	70	128	0.290500	298351/1027026	$13, 12, 13, 11, 9, 7, 5$	$6, 5, 5, 4, 3, 2, 1$
7	71	130	0.290163	1211191/4174170	$14, 12, 13, 11, 9, 7, 5$	$6, 5, 5, 4, 3, 2, 1$
7	72	132	0.289832	1194889/4122690	$14, 12, 13, 11, 9, 8, 5$	$6, 5, 5, 4, 3, 2, 1$

5.5.4 Conclusions

We have modified Theorem 5.2.1 to solve the problem of finding divisors in residue classes $(sx + r)|N$, and used this to bound the number of such divisors for a given $s = N^\alpha$, $\alpha > 1/4$ by $c(\alpha) = O((\alpha - 1/4)^{-3/2})$. In fact Lenstra's technique can also be extended to show that $c(\alpha) = O((\alpha - 1/4)^{-3/2})$, as explained in (Coppersmith *et al.*, 1998).

We then worked out the values of $c(\alpha)$ for all $\alpha > 0.29$; as shown in the final table of the last section. Previously known results due to H. W. Lenstra for $\alpha \geq 1/3$ are as follows:

$\alpha >$	$c(\alpha) \leq$
$1/2 = 0.5$	2
$2/5 = 0.4$	4
$3/8 \approx 0.375$	6
$4/11 \approx 0.364$	7
$13/37 \approx 0.351$	8
$9/26 \approx 0.346$	9
$31/92 \approx 0.337$	10
$1/3 \approx 0.333$	11

Lenstra's bounds can be seen to be considerably better. Firstly this is thought to be true because the lattice methods find the divisors of N of the form $sx - r$ as well (it is not presently known how to seperate the two problems with lattices). However Lenstra's results are over twice as good which might imply there is another factor that needs to be taken in to account. It would be very interesting to try to align these two sets of results.

In contrast to the given problem one can consider the question of how to construct numbers N with a given number of divisors in the same residue class. Cohen has shown that there are an infinitely many numbers with 6 divisors in the same residue class, i.e. those of the form $n = (2x+1)(x^2+1)(x^2+x+1)(2x^2-x+1)(2x^2+x+1)$ with $r = 1$ and $s = (2x+1)(x^2+1) - 1$. Since $s > N^{1/3}$ for all $x > 5$ we have that $c(1/3) \geq 6$. Also $n = (x+1)(2x+1)$ with $s = x$ and $r = 1$ shows that $c(\alpha) \geq 4$ for all $\alpha < 1/2$.

It seems that a considerable amount of further work is needed to join the two problems, and therefore have exact upper bounds on the number of divisors in residue classes for given $\alpha > 1/4$.

Part IV:
Wiener-type attacks on RSA

Chapter 6

Wiener-type attacks on RSA

For efficient RSA signature generation it may be tempting to use a small private exponent d. Unfortunately, Wiener (Wiener, 1990) has shown that when the RSA protocol is used with a decrypting exponent, d, less than $N^{1/4}$ and an encrypting exponent, e, approximately the same size as N, then the RSA system can be broken[1] in time polynomial in $d/N^{1/4}$ (see Section 6.1.1). Very recently Boneh and Durfee (Boneh & Durfee, 1999) managed to improve Wiener's result by showing how to break the RSA cryptosystem even when using decrypting exponents of size up to $N^{0.292}$; their idea is described in Section 6.1.3.

In order to simplify the RSA key management one may also be tempted to use a single modulus for several key pairs e_i, d_i. However, as pointed out by Simmons (Simmons, 1983), whenever a message m is sent to two participants whose public exponents happen to be relatively prime, then the message m can be easily recovered without breaking the system. DeLaurentis (DeLaurentis, 1984) described two further attacks in which a participant can break such a common modulus cryptosystem. Particularly, he showed that knowledge of one key pair e_i, d_i gives rise to an efficient probabilistic algorithm for factoring the modulus N. Moreover, he also showed that knowledge of one key pair e_i, d_i gives rise to an efficient deterministic algorithm to generate other key pairs without determining $\lambda(N)$. For a thorough discussion of the common modulus situation when using RSA we refer to Moore (Moore, 1992). However note that Simmons's attack does not break the RSA system at all and the attack of DeLaurentis assumes that the attacker is also given the secret exponent.

[1] We have already seen one way of extending this attack in Section 5.5.1; when a low public exponent is being used and just $1/4$ of the least significant bits of d have been revealed (rather than knowing the top $3/4$ of the bits of d are all zero).

In this chapter we study the more realistic problem of what an opponent might do, given only several public exponents for a given modulus and the knowledge that the corresponding private exponents are quite small. Such a position could possibly occur if a person is using the same modulus N, but different exponents e_i to sign different classes of message.

Even though this situation is not common in present-day RSA systems, an analysis of the problem sheds some light on the gain of additional public information in attacking RSA and on the security of re-using the modulus N. Moreover, it is an interesting mathematical problem in its own right, for which the lattice based solution shown below is an elegant solution. The general technique used here may also be useful in other circumstances. Finally if it is true, as hypothesised in (Boneh & Durfee, 1999), that Wiener's attack can be extended up to $N^{1/2-\varepsilon}$ then perhaps this technique could also be improved in a similar way. If one were to assume this, then it could possibly restrict the search for a solution to extending Wiener's original attack.

The question of how to combine several public exponents for a given modulus in order to reduce the size constraint on the private exponents for their efficient reconstruction was first studied by Guo (Guo, 1996). Still based on the continued fraction approach of Wiener, Guo showed how to break RSA given 3 public exponents even when their corresponding decrypting exponents are of size less than $N^{1/3}$. This method is described in Section 6.1.2. Using instead a lattice basis reduction approach we continue this study in Section 6.2, generalising (and improving) the result up to an arbitrary number of exponents. Particularly, we show that with n encrypting exponents e_i, the lattice basis approach allows for the d_i to be as large as N^{α_n} where

$$\alpha_n = \begin{cases} \frac{(2n+1)2^n-(2n+1)\binom{n}{n/2}}{(2n-2)2^n+(4n+2)\binom{n}{n/2}} & \text{if } n \text{ is even,} \\ \\ \frac{(2n+1)2^n-4n\binom{n-1}{(n-1)/2}}{(2n-2)2^n+8n\binom{n-1}{(n-1)/2}} & \text{if } n \text{ is odd.} \end{cases}$$

It is interesting to note that the method of Section 6.2 allows for 2 encrypting exponents a decrypting exponent bound of $N^{5/14}$, which is superior to the $N^{1/3}$ bound of Guo even for 3 encrypting exponents.

The author is grateful for helpful cooperation with Jean-Pierre Seifert in the work of this chapter.

6.1 Low private exponent attacks on RSA

6.1.1 Wiener's approach

It was shown in Wiener (Wiener, 1990) that, if one assumes $\lambda(N)$ and e are both approximately as large as N, and if the decrypting exponent d is less than $N^{1/4}$ then the modulus N can be factored by examining the continued fraction approximation of e/N. This follows because e and d satisfy the relationship $ed - k\lambda(N) = 1$. So letting $\lambda(N) = (p-1)(q-1)/g$, and $s = 1 - p - q$ we have that

$$edg - kN \quad = \quad g + ks. \tag{6.1}$$

Dividing both sides by dgN gives

$$\frac{e}{N} - \frac{k}{dg} \;=\; \frac{g+ks}{dgN} \;=\; \left(\frac{k}{dg}\right)\left(\frac{s}{N}\right) + \frac{1}{dN}.$$

Now using the assumption that $e \simeq N$, and that $s \simeq N^{1/2}$ means (from examining equation 6.1) that $k/(dg) \simeq 1$ so that the right-hand side of the above equation is approximately $N^{-1/2}$. It is well known (see for instance (Hardy & Wright, 1979)) that if

$$|x - a/b| \quad < \quad 1/(2b^2)$$

then a/b is a continued fraction approximant of x. Thus if $N^{-1/2} < 1/(2(dg)^2)$ then $k/(dg)$ will be a continued fraction approximant of e/N. This is true whenever

$$d \quad < \quad 2^{-1/2}(1/g)N^{1/4}, \tag{6.2}$$

and g will be small under the assumption that $\lambda(N) \simeq N$ (though clearly $g \geq 2$ since both p and q are odd). Given k/dg one may calculate

$$r \;=\; (p-1)(q-1) \;=\; \frac{edg}{k} - \frac{g}{k} \;=\; \lfloor edg/k \rceil \qquad \text{(since } g \text{ is small)},$$

and then we can factor N since the factors p and q satisfy the quadratic relationship $x^2 - (N+1-r)x + N = 0$.

6.1.2 Guo's approach

The approach taken in Guo (Guo, 1996) assumes that one has more than one e_i for a given N, and that each of these e_i has a relatively small d_i. Guo only considers the problem for 2 and 3 encryption exponents. For 2 exponents we have the following relations:

$$\begin{aligned} e_1 d_1 g - k_1(p-1)(q-1) &= g \\ e_2 d_2 g - k_2(p-1)(q-1) &= g, \end{aligned}$$

so multiplying the first by k_2, the second by k_1, and subtracting gives

$$k_2 d_1 e_1 - k_1 d_2 e_2 = k_2 - k_1. \tag{6.3}$$

Dividing now both sides of equation 6.3 by $k_2 d_1 e_2$ implies the following

$$\frac{e_1}{e_2} - \frac{k_1 d_2}{k_2 d_1} = \frac{k_2 - k_1}{k_2 d_1 e_2},$$

and assuming that the d_i (and hence k_i if the e_i are large) are at most N^{α} means that the right-hand side is about $N^{-(1+\alpha)}$.

For the fraction $k_1 d_2/(k_2 d_1)$ to be a continued fraction approximant of e_1/e_2, we must therefore have that

$$2(k_2 d_1)^2 < N^{1+\alpha},$$

and with the assumptions that k_2 and d_1 are at most N^{α} and that g is small this condition will be true whenever $\alpha = 1/3 - \epsilon$ for some $\epsilon > 0$.

However, unlike Wiener's attack, the fraction $k_1 d_2/(k_2 d_1)$ does not break the RSA cryptosystem for two reasons:

- Firstly knowing, say, the numerator $k_1 d_2$, does not allow us to find d_2 or k_1 without factoring this number.
- Secondly there may be a factor in common between $d_1 k_2$ and $d_2 k_1$ in which case the continued fraction method would not give a fraction with numerator $k_1 d_2$ and denominator $k_2 d_1$, but rather the fraction with the common factor removed.

Guo assumes that the second problem does not exist, i.e. that we have $\gcd(k_1 d_2, k_2 d_1) = 1$, and it is estimated that this happens with probability $6/\pi^2 \simeq 0.61$.

To get around the first problem, Guo suggests that one could either try to factor $k_1 d_2$ (a number of size about $N^{2/3}$ and not typically of a hard factorisation shape), or alternatively assume that one has another encrypting exponent e_3 with $d_3 < N^{1/3}$. Then (repeating the above procedure with e_3 and e_2) one can also find $k_3 d_2$, and calculating $\gcd(k_1 d_2, k_3 d_2)$ will hopefully (if $\gcd(k_1, k_3) = 1$) give d_2 and thus allow the factoring of N. The probability of this attack working under the given assumptions is $(6/\pi^2)^3 \simeq 0.23$.

6.1.3 Boneh and Durfee's approach

For simplicity assume that $\gcd(p-1, q-1) = 2$. Then to break RSA, and indeed to factor N we must find an (x, y, z) solution to the equation

$$x(m + y) + ze = 1,$$

where $m = (N+1)/2$, $y = -(p+q)/2$, z is the decrypting exponent d and x is the other coefficient from the extended Euclidean algorithm (around the same size as z).

Boneh and Durfee (see (Boneh & Durfee, 1999)) named this "the small inverse problem". In their approach they treated this equation modulo e, and then used the univariate modular approach of Section 4.1 on the resulting bivariate integer equation. The advancement was multiplying by both x and y in forming the necessary polynomial relationships for LLL to reduce.

With a certain, natural choice of polynomials they improved the bound on the susceptible d (i.e. d which lead to a polynomial time factoring of N) to N^α where $\alpha = (7 - 2\sqrt{7})/6 \approx 0.285$. Then, by removing some of the "less good" relations, they were able to further improve this to $\alpha = (2 - \sqrt{2})/2 \approx 0.292$. See (Boneh & Durfee, 1999) for details.

6.2 An extension in the presence of many small decryption exponents

In this section we will use, among others, ideas from both Wiener and Guo to solve the general problem of breaking RSA in the presence of n encrypting exponents e_i, all with relatively small $d_i < N^{\alpha_n}$, $i = 1, \ldots, n$. The main technique used in deriving these results is the creation and subsequent reduction of certain lattices. The approach taken can currently only be classed as a heuristic method because, although the vectors we search for can be shown to be relatively short, we cannot yet prove that they are bound to be found by lattice basis reduction algorithms. Nevertheless, in Section 6.3 it is shown that our approach performs well in practice, and that the following theoretically derived bounds are frequently achieved. In particular, in the presence of n encrypting exponents e_i, our approach allows for the d_i to be as large as N^{α_n} where

$$\alpha_n = \begin{cases} \frac{(2n+1)2^n-(2n+1)\binom{n}{n/2}}{(2n-2)2^n+(4n+2)\binom{n}{n/2}} & \text{if } n \text{ is even,} \\ \\ \frac{(2n+1)2^n-4n\binom{n-1}{(n-1)/2}}{(2n-2)2^n+8n\binom{n-1}{(n-1)/2}} & \text{if } n \text{ is odd.} \end{cases}$$

The first few (from $n = 1$) start 1/4, 5/14, 2/5, 15/34, 29/62. In Section 6.2.5 it is shown that $\alpha_n \to 1$ as $n \to \infty$.

If the LLL algorithm (see (Lenstra *et al.*, 1982)) is used in order to reduce the lattices underlying our approach, and the (pessimistic) estimate for its complexity of $O(m^6 \log^3 B)$ is assumed (given a lattice of dimension m with largest norm B), then the complexity of our method is $O(2^{6n}n^3 \log^3 N)$, and so clearly the attack is only practical for small n.

6.2.1 Preliminaries

In extending the analysis to n encrypting exponents e_i (with small decrypting exponents d_i), we use both Wiener's and Guo's ideas. We shall refer to relations of the form

$$d_i g e_i - k_i N = g + k_i s$$

as Wiener equations, and we shall denote them by W_i (see equation 6.1 for an example). Similarly we shall refer to relations of the form

$$k_i d_j e_j - k_j d_i e_i = k_i - k_j$$

as Guo equations, and shall denote them $G_{i,j}$ (see equation 6.3 for an example). We shall also assume, for a given n, that the d_i and k_i are at most N^{α_n}, that g is small, and that s is around $N^{1/2}$. Notice that the right-hand sides of W_i and $G_{i,j}$ are therefore quite small; in fact at most $N^{(1/2)+\alpha_n}$, and N^{α_n} respectively. Finally we often refer to composite relations, e.g. $W_u G_{v,w}$, in which case we mean the relation, whose left-hand (resp. right-hand) side is the product of the left-hand (resp. right-hand) sides of W_u and $G_{v,w}$. For example, $W_u G_{v,w}$ which has a relatively small right-hand side, bounded in size by $N^{(1/2)+2\alpha_n}$.

In the following analysis we examine the cases of 2, 3 and 4 exponents before generalising the approach to n exponents. This is done both to give explicit examples of the approach when in the presence of a small number of exponents, and also because it is not until the presence of 4 exponents that the general phenomenon becomes clear. The relations that we choose for the cases of 2, 3 and 4 exponents may seem "plucked from the air", but the pattern is made clear in Section 6.2.5.

6.2.2 RSA in the presence of 2 small decryption exponents

Assuming that we have two small decryption exponents, then the following relations hold: $W_1, G_{1,2}, W_1W_2$; or more explicitly:

$$\begin{aligned}
d_1ge_1 - k_1N &= g + k_1s, \\
k_1d_2e_2 - k_2d_1e_1 &= k_1 - k_2, \\
d_1d_2g^2e_1e_2 - d_1gk_2e_1N - d_2gk_1e_2N + k_1k_2N^2 &= (g + k_1s)(g + k_2s).
\end{aligned}$$

Multiplying the first of these by k_2 and the second by g means that the left-hand sides are all in terms of $d_1d_2g^2$, d_1gk_2, d_2gk_1, and k_1k_2, and hence we may write these equations in the matrix form below.

$$(k_1k_2, d_1gk_2, d_2gk_1, d_1d_2g^2) \quad \begin{pmatrix} 1 & -N & 0 & N^2 \\ & e_1 & -e_1 & -e_1N \\ & & e_2 & -e_2N \\ & & & e_1e_2 \end{pmatrix} =$$

$$(k_1k_2, k_2(g + k_1s), g(k_1 - k_2), (g + k_1s)(g + k_2s).$$

The size of the entries of the vector on the right-hand side are at most $N^{2\alpha_2}$, $N^{(1/2)+2\alpha_2}$, N^{α_2}, and $N^{1+2\alpha_2}$ respectively. These size estimates may be made roughly equivalent by multiplying the first three columns of the matrix by N, $M_1 = N^{1/2}$, and $M_2 = N^{1+\alpha_2}$ respectively, which gives the following matrix:

$$L_2 = \begin{pmatrix} N & -M_1N & 0 & N^2 \\ & M_1e_1 & -M_2e_1 & -e_1N \\ & & M_2e_2 & -e_2N \\ & & & e_1e_2 \end{pmatrix}$$

In this case the vector $b = (k_1k_2, d_1gk_2, d_2gk_1, d_1d_2g^2)$ will be such that

$$\|bL_2\| < 2N^{1+2\alpha_2}.$$

We must now make the assumption that, in the lattice generated by the rows of L_2, the shortest vector has length $\Delta^{1/4-\epsilon}$, where $\Delta = \det(L_2) \approx N^{(13/2)+\alpha_2}$, and moreover that the next shortest linearly independent vector has a significantly larger norm than the shortest vector in L_2. Indeed, if the lattice L_2 is pretty "random", there are almost surely no lattice points of L_2 significantly shorter than the Minkowski bound $2\Delta^{1/4}$. Under these assumptions, then bL_2 is the shortest vector in the lattice if

$$N^{1+2\alpha_2} \quad < \quad (1/c_2)\left(N^{(13/2)+\alpha_2}\right)^{1/4}$$

for some small c_2, which is true if

$$\alpha_2 < 5/14 - \epsilon'.$$

This implies that the vector $b = (b_1, b_2, b_3, b_4)$ can be found via lattice basis reduction algorithms (e.g. LLL) if $\alpha_2 < 5/14 - \epsilon'$, and then $d_1g/k_1 = b_2/b_1$ can be calculated, which leads to the factoring of N as shown in Section 6.1.1.

6.2.3 RSA in the presence of 3 small decryption exponents

This method extends easily to 3 encrypting exponents. We now have the quantities $1, e_1, e_2, e_1e_2, e_3, e_1e_3, e_2e_3$ and $e_1e_2e_3$ from which to form linear relationships, and we already have relationships concerning the first four of these from the 2 exponent case, namely $1, W_1, G_{1,2}$ and W_1W_2. For the remaining relationships we choose $G_{1,3}$, $W_1G_{2,3}$, $W_2G_{1,3}$ and $W_1W_2W_3$. These relations imply looking for the vector

$$\begin{aligned} b \quad = \quad & (k_1k_2k_3, d_1gk_2k_3, k_1d_2gk_3, d_1d_2g^2k_3, \\ & k_1k_2d_3g, k_1d_3g, k_2d_3g, d_1d_2d_3g^3), \end{aligned}$$

by reducing the rows of the following lattice:

$$L_3 = \begin{pmatrix} 1 & -N & 0 & N^2 & 0 & 0 & 0 & -N^3 \\ & e_1 & -e_1 & -e_1N & -e_1 & 0 & e_1N & e_1N^2 \\ & & e_2 & -e_2N & 0 & e_2N & 0 & e_2N^2 \\ & & & e_1e_2 & 0 & -e_1e_2 & -e_1e_2 & -e_1e_2N \\ & & & & e_3 & -e_3N & -e_3N & e_3N^2 \\ & & & & & e_1e_3 & 0 & -e_1e_3N \\ & & & & & & e_2e_3 & -e_2e_3N \\ & & & & & & & e_1e_2e_3 \end{pmatrix} \times D,$$

where D is the diagonal matrix

$$\mathrm{diag}(N^{3/2}, N, N^{(3/2)+\alpha_3}, N^{1/2}, N^{(3/2)+\alpha_3}, N^{1+\alpha_3}, N^{1+\alpha_3}, 1)$$

used to maximise the determinant of L_3 and still keep

$$||bL_3|| < \sqrt{8}N^{(3/2)+3\alpha_3}.$$

Again, using the assumptions that the shortest vector in the lattice generated by the rows of L_3 has length $\det(L_3)^{(1/8)-\epsilon}$, and is also significantly shorter than the next shortest linearly independent vector in L_3, means that bL_3 will be the shortest vector in the lattice L_3 if

$$N^{(3/2)+3\alpha_3} \quad < \quad (1/c_3)\left(N^{20+4\alpha_3}\right)^{1/8}$$

for some small c_3 which is true if

$$\alpha_3 < 2/5 - \epsilon'.$$

By using again the first two components of b, as in the 2 exponent case, one may now factor the modulus N as shown in Section 6.1.1.

6.2.4 RSA in the presence of 4 small decryption exponents

In the presence of 4 exponents we can now use linear relationships among the quantities 1, e_1, e_2, e_1e_2, e_3, e_1e_3, e_2e_3, $e_1e_2e_3$, e_4, e_1e_4, e_2e_4, e_3e_4, $e_1e_2e_4$, $e_1e_3e_4$, $e_2e_3e_4$ and $e_1e_2e_3e_4$. As before we already have linear relationships for the first half of these quantities from the analysis in the presence of 3 equations. For the remaining quantities we use the relations $G_{1,4}$, $W_1G_{2,4}$, $G_{1,2}G_{3,4}$, $G_{1,3}G_{2,4}$, $W_1W_2G_{3,4}$, $W_1W_3G_{2,4}$, $W_2W_3G_{1,4}$ and $W_1W_2W_3W_4$. Putting these relations in matrix form, and multiplying the columns by appropriate factors to make all the relations of size at most $N^{2+4\alpha_4}$, results in a 16×16 matrix, L_4, which has determinant $N^{(109/2)+13\alpha_4}$. The vector b we are now looking for is

$$\begin{aligned} b \quad = \quad & (k_1k_2k_3k_4, d_1gk_2k_3k_4, k_1d_2gk_3k_4, d_1d_2g^2k_3k_4, \\ & k_1k_2d_3gk_4, d_1k_2d_3g^2k_4, k_1d_2d_3g^2k_4, d_1d_2d_3g^3k_4, \\ & k_1k_2k_3d_4g, d_1k_2k_3d_4g^2, k_1d_2k_3d_4g^2, k_1k_2d_3d_4g^2, \\ & d_1d_2k_3d_4g^3, d_1k_2d_3d_4g^3, k_1d_2d_3d_4g^3, d_1d_2d_3d_4g^4). \end{aligned}$$

Therefore, again making the same assumptions as before, implies that the vector bL_4 is the shortest vector in the lattice generated by the rows of L_4 if

$$N^{2+4\alpha_4} \quad < \quad (1/c_4)\left(N^{(109/2)+13\alpha_4}\right)^{1/16}$$

for some small c_4, and this is true if

$$\alpha_4 < 15/34 - \epsilon'.$$

Using again the first two components of b, as in the 2 and 3 exponent case, one may again factor the modulus N as shown in Section 6.1.1.

6.2.5 The general approach

We now work out the general bound on the d_i when we have n encrypting exponents. The reader is encouraged to refer back to the previous sections (when $n = 2, 3$ and 4) as examples.

Given that there are n exponents e_i, then there are 2^n different quantities, h_j, involving the e_i's, and the product of all of these (assuming $e \approx N$) is $N^{(n2^{n-1})}$. This means that one considers a diagonal matrix, L_n, of dimension 2^n, and that the determinant of this matrix, before multiplying the rows to increase the allowable bound, is $N^{(n2^{n-1})}$.

The last relation $W_1W_2 \dots W_n$ has a right-hand side of at most $N^{(n/2)+n\alpha_n}$, and thus we increase the right-hand side of all the other relations up to this bound, making the desired vector b such that $\|bL_n\|_\infty$ is (still) approximately $N^{(n/2)+n\alpha_n}$. The general form of the desired vector b is that its j^{th} entry is the product of n unknown quantitities a_i for $i = 1 \dots n$, where a_i is either d_ig or k_i depending on whether e_i is present in the

j^{th} quantity h_j or not.

We now consider the interesting problem of which relations to consider for n equations. Observe that a general relation of the form

$$R_{u,v} \quad = \quad W_{i_1} \ldots W_{i_u} G_{j_1,l_1} \ldots G_{j_v,l_v},$$

(where the $i_1, \ldots, i_u, j_1, \ldots, j_v, l_1, \ldots, l_v$ are unique), has a left-hand side composed of products of $(u + 2v)$ of the e_i's with coefficients that are products of $(u + v)$ of the unknown quantities a_i (where a_i is again either $d_i g_i$ or k_i). Also notice that the right-hand side of $R_{u,v}$ has size at most $N^{(u/2)+(u+v)\alpha_n}$.

Our method requires all the coefficients to be roughly the same size (a product of n of the quantities a_i). This means that relations which have coefficients less than this must be multiplied (on both sides) by some missing k_i. For example, in the the 2 exponent case we multiplied the first equation by k_2 to make all the coefficients of size $N^{2\alpha_2}$. This has the effect of increasing the right-hand side of relation $R_{u,v}$ to a size bounded by $N^{(u/2)+(n-v)\alpha_n}$.

Given this new relation $R_{u,v}$ we now need to make its right-hand side as large as the right-hand side of $W_1 W_2 \ldots W_n$, which means multiplying (both sides) by $N^{(n-u)/2+v\alpha_n}$. For example, these multiplication factors are the (diagonal) entries of the diagonal matrix D in the example when $n = 3$.

Say that the product of these multiplication factors (i.e. the determinant of D in the $n = 3$ example) is N^{β_n}, where $\beta_n = x + y\alpha_n$, and let L_n denoted the lattice of (modified) relations as before. This means that (under the usual assumptions) the vector bL_n is the shortest vector of the lattice if

$$N^{n/2+n\alpha_n} \quad < \quad (1/c_n)\left(N^{n2^{n-1}+x+y\alpha_n}\right)^{1/2^n}$$

for some small c_n, i.e. when

$$\alpha_n \quad < \quad \frac{x}{n2^n - y} - \epsilon'. \tag{6.4}$$

In order to maximise α_n we wish both x and y to be large. This means that the relations should be chosen to maximise v (and minimise u). For instance when $n = 2$ we choose the relations $W_1, G_{1,2}$ and $W_1 W_2$ rather than W_1, W_2 and $W_1 W_2$ because $\beta_2 = 2$ in the latter case rather than $5/2 + \alpha_2$ in the former.

With this general principle in mind we still need to explain exactly which relations we use. In order to mantain the triangularity of L_n we only consider relations which introduce one new quantity h_j. The choices for $n \leq 5$ can be seen in the below figure.

h_j	relation	size of coeffs	size of h_j	size of rhs	contribution to β_n
1	–	0	0	0	$(n/2)$
e_1	W_1	1	1	$(1/2) + \alpha_n$	$(n-1)/2$
e_2	$G_{1,2}$	2	1	α_n	$(n/2) + \alpha_n$
e_1e_2	W_1W_2	2	2	$1 + 2\alpha_n$	$(n-2)/2$
e_3	$G_{1,3}$	2	1	α_n	$(n/2) + \alpha_n$
e_1e_3	$W_1G_{2,3}$	3	2	$(1/2) + 2\alpha_n$	$(n-1)/2 + \alpha_n$
e_2e_3	$W_2G_{1,3}$	3	2	$(1/2) + 2\alpha_n$	$(n-1)/2 + \alpha_n$
$e_1e_2e_3$	$W_1W_2W_3$	3	3	$(3/2) + 3\alpha_n$	$(n-3)/2$
e_4	$G_{1,4}$	2	1	α_n	$(n/2) + \alpha_n$
e_1e_4	$W_1G_{2,4}$	3	2	$(1/2) + 2\alpha_n$	$(n-1)/2 + \alpha_n$
e_2e_4	$G_{1,2}G_{3,4}$	4	2	$2\alpha_n$	$(n/2) + 2\alpha_n$
e_3e_4	$G_{1,3}G_{2,4}$	4	2	$2\alpha_n$	$(n/2) + 2\alpha_n$
$e_1e_2e_4$	$W_1W_2G_{3,4}$	4	3	$1 + 3\alpha_n$	$(n-2)/2 + \alpha_n$
$e_1e_3e_4$	$W_1W_3G_{2,4}$	4	3	$1 + 3\alpha_n$	$(n-2)/2 + \alpha_n$
$e_2e_3e_4$	$W_2W_3G_{1,4}$	4	3	$1 + 3\alpha_n$	$(n-2)/2 + \alpha_n$
$e_1e_2e_3e_4$	$W_1W_2W_3W_4$	4	4	$2 + 4\alpha_n$	$(n-4)/2$
e_5	$G_{1,5}$	2	1	α_n	$(n/2) + \alpha_n$
e_1e_5	$W_1G_{2,5}$	3	2	$(1/2) + 2\alpha_n$	$(n-1)/2 + \alpha_n$
e_2e_5	$G_{1,2}G_{3,5}$	4	2	$2\alpha_n$	$(n/2) + 2\alpha_n$
e_3e_5	$G_{1,3}G_{4,5}$	4	2	$2\alpha_n$	$(n/2) + 2\alpha_n$
e_4e_5	$G_{1,4}G_{2,5}$	4	2	$2\alpha_n$	$(n-2)/2 + \alpha_n$
$e_1e_2e_5$	$W_1W_2G_{4,5}$	4	3	$1 + 3\alpha_n$	$(n-1)/2 + 2\alpha_n$
$e_1e_3e_5$	$W_1G_{2,3}G_{4,5}$	5	3	$(1/2) + 3\alpha_n$	$(n-1)/2 + 2\alpha_n$
$e_1e_4e_5$	$W_1G_{2,4}G_{3,5}$	5	3	$(1/2) + 3\alpha_n$	$(n-1)/2 + 2\alpha_n$
$e_2e_3e_5$	$W_2G_{1,3}G_{4,5}$	5	3	$(1/2) + 3\alpha_n$	$(n-1)/2 + 2\alpha_n$
$e_2e_4e_5$	$W_2G_{1,4}G_{3,5}$	5	3	$(1/2) + 3\alpha_n$	$(n-1)/2 + 2\alpha_n$
$e_3e_4e_5$	$W_3G_{2,4}G_{1,5}$	5	3	$(1/2) + 3\alpha_n$	$(n-1)/2 + 2\alpha_n$
$e_1e_2e_3e_5$	$W_1W_2W_3G_{4,5}$	5	4	$(3/2) + 4\alpha_n$	$(n-3)/2 + \alpha_n$
$e_1e_2e_4e_5$	$W_1W_2W_4G_{3,5}$	5	4	$(3/2) + 4\alpha_n$	$(n-3)/2 + \alpha_n$
$e_1e_3e_4e_5$	$W_1W_3W_4G_{2,5}$	5	4	$(3/2) + 4\alpha_n$	$(n-3)/2 + \alpha_n$
$e_2e_3e_4e_5$	$W_2W_3W_4G_{1,5}$	5	4	$(3/2) + 4\alpha_n$	$(n-3)/2 + \alpha_n$
$e_1e_2e_3e_4e_5$	$W_1W_2W_3W_4W_5$	5	5	$(5/2) + 5\alpha_n$	$(n-5)/2$

A table showing the chosen relations for $n \leq 5$.

After the initial "base relation" (which requires that the first component of b should be small), we seek a linear relation between e_1 and 1 (or a multiple of this e.g. N), and our only choice for this is W_1. With the introduction of the next exponent e_2 we now look for a relation between $1, e_1$ and e_2. For this we can either choose W_2 or $G_{1,2}$, and as explained above $G_{1,2}$ is the right choice.

A more interesting situation arises when the fourth exponent e_4 has been introduced, and one looks for a relation regarding e_1e_4 and the previous ones. The best choice in this case turns out to be $W_1G_{2,4}$. However, when considering the next relation regarding e_2e_4 and the previous ones we may now use $G_{1,2}G_{3,4}$ because the left-hand side of this relation contains e_1e_3, e_1e_4, e_2e_3 and e_2e_4 all of which are now present.

In general when looking for a relation regarding $e_{i_1}e_{i_2}\dots e_{i_s}$ and the previous ones, one can use any relation $R_{u,v}$ where $u+v=s$, subject to the required h_j being present earlier. It can be shown that the number of relations R_{u+v} with $v=t$ should be $\binom{n}{t}-\binom{n}{t-1}$ regardless of the size $s=u+v$ of the relation (though of course this is subject to $t\le s$ and $s+2t\le n$). The contribution to β_n for such a relation is $(n-s+t)/2+t\alpha_n$, and thus (summing over the possible n) the total contribution to β_n is shown below.

$$\beta_n = \sum_{s=0}^{n}\sum_{t=0}^{\min(s,n-s)}\left(\binom{n}{t}-\binom{n}{t-1}\right)\left(\frac{n-s+t}{2}+t\alpha_n\right)$$

Assuming n is even this sum can be simplified to

$$\beta_n = \frac{(2n+1)2^n-(2n+1)\binom{n}{n/2}}{4}+\frac{(n+1)2^n-(2n+1)\binom{n}{n/2}}{2}\,\alpha_n,$$

or if n is odd then the sum becomes

$$\beta_n = \frac{(2n+1)2^n-4n\binom{n-1}{(n-1)/2}}{4}+\frac{(n+1)2^n-4n\binom{n-1}{(n-1)/2}}{2}\,\alpha_n.$$

Using equation 6.4 this means that if n is even, then

$$\alpha_n = \frac{(2n+1)2^n-(2n+1)\binom{n}{n/2}}{(2n-2)2^n+(4n+2)\binom{n}{n/2}}, \tag{6.5}$$

whilst if n is odd, then

$$\alpha_n = \frac{(2n+1)2^n-4n\binom{n-1}{(n-1)/2}}{(2n-2)2^n+8n\binom{n-1}{(n-1)/2}}. \tag{6.6}$$

Either way, using Stirling's formula $n! \approx \sqrt{2\pi n} n^n e^{-n}$ we have that

$$\binom{2k}{k} = \frac{(2k)!}{(k!)^2} \approx \frac{1}{\sqrt{\pi k}} 2^{2k} \ll 2^{2k}$$

as $k \to \infty$, which shows that $\alpha_n \to 1$ as $n \to \infty$.

The theoretical limits for which α_n when n varies from 1 to 100 are shown below for completeness sake, and to show how slowly the curve tends to 1. However one should understand that for n above 10 say, the method is completely infeasibly due to the size of the matrix to be reduced.

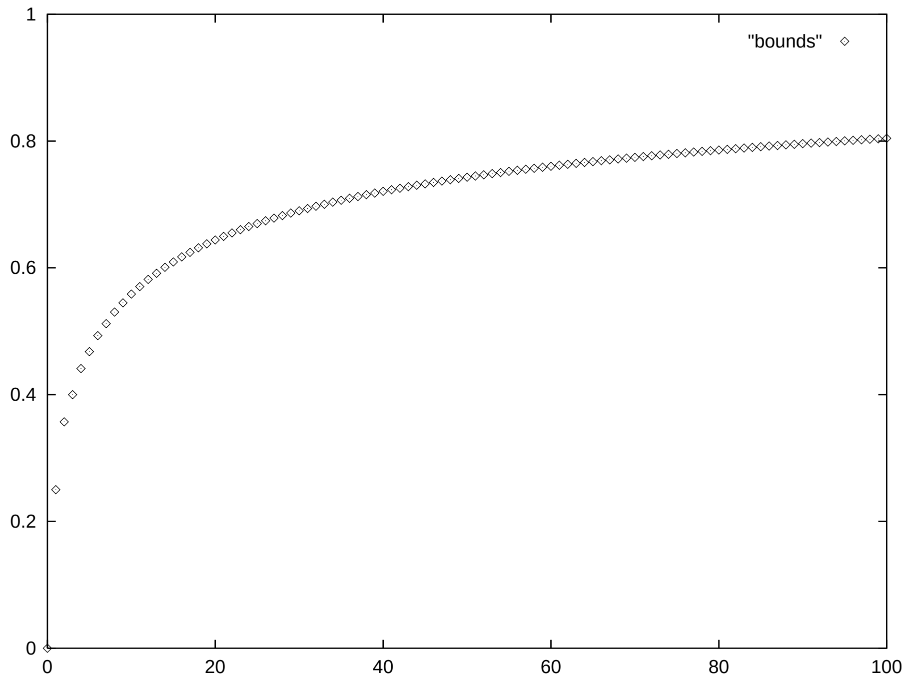


Fig. 1. Graph of bounds α_n for $n \leq 100$.

6.3 Practical results

Although our method is at the current time only heuristic, it works well in practice as can be seen from our experimental results below.

Our implementation uses the NTL library (Shoup, 1995) of Victor Shoup. Timings are given for a 300 MHz AMD K6 running under Linux.

Table 6.3.1 Average running time (in seconds) and success rate for 10 random experiments as a function of α_2.

RSA-500 with 2 public exponents			
α_2	bit length of d_i	avg. time in secs.	success rate
0.356	178	0.441	40%
0.354	177	0.421	100%

Table 6.3.2 Average running time (in seconds) and number of success rate for 10 random experiments as a function of α_2.

RSA-700 with 2 public exponents			
α_2	bit length of d_i	avg. time in secs.	success rate
0.357143	250	1.075	0%
0.355714	249	1.117	70%
0.354286	248	0.93	80%
0.352857	247	1.33	100%

Table 6.3.3 Average running time (in seconds) and success rate for 10 random experiments as a function of α_3.

RSA-500 with 3 public exponents			
α_3	bit length of d_i	avg. time in secs.	success rate
0.4	200	3.632	0%
0.398	199	3.567	40%
0.396	198	3.599	90%
0.394	197	3.726	90%
0.392	196	3.595	90%
0.39	195	3.529	100%

Table 6.3.4 Average running time (in seconds) and success rate for 10 random experiments as a function of α_4.

RSA-200 with 4 public exponents			
α_4	bit length of d_i	avg. time in secs.	success rate
0.44	88	14.538	0%
0.435	87	14.496	50%
0.43	86	14.328	80%
0.425	85	14.159	100%

Table 6.3.5 Average running time (in seconds) and success rate for 10 random experiments as a function of α_5.

RSA-200 with 5 public exponents			
α_5	bit length of d_i	avg. time in secs.	success rate
0.45	90	424.756	0%
0.445	89	427.275	60%
0.44	88	422.74	100%

6.4 Open problems

In (Wiener, 1990) it was commented that the attack could not be brought to bear if the Chinese remainder theorem variant of RSA was being employed, i.e. d was not small, but $d_p = d \pmod{p-1}$ and $d_q = d \pmod{q-1}$ are both small (and hence decryption is still speeded up). One should observe that the generalised problem (with many e_i and d_i) seems equally flawed against exposing this choice of RSA exponents.

Another theoretical problem raised by this work is to take steps towards proving the lattice assumptions used in Section 6.2. As the experimental results strongly support the derived bounds it is natural to ask whether the attack can be turned into a rigorous theorem.

Lastly, in (Boneh & Durfee, 1999), Boneh and Durfee conjectured that there may be a polynomial time solution for the small inverse problem, for any d as large as $N^{1/2-\varepsilon}$. It would seem that we are presently lacking techniques capable of doing this, but if this result is proved, it would be very interesting to see if the techniques of Section 6.2 could be used to generalise it to many exponents e_i and d_i.

Part V:
Conclusions and Open Problems

Chapter 7

Conclusions and open problems

Below is a brief summary of each chapter of the thesis for the reader to recap. The main points of the thesis are highlighted in the following sections: the first of these shows the new results that have been achieved from this work, whilst the second discusses the problems that remain open.

A brief summary of the thesis

Chapter 1 gives an introduction to cryptography, and then introduces each of the subsequent chapters in detail. See Section 1.3 for a more detailed account of each of the chapters below.

Chapter 2 considers the study of smooth numbers in relation to detecting torsion in a group, and shows that artificial smoothness constraints make this problem easier than may be thought.

Chapter 3 then gives a solid introduction to the theory of lattices, proving numerous results which are used in subsequent chapters.

Chapter 4 studies the problem of finding small solutions to univariate modular equations. It gives an alternative (and simpler) method to that proposed in (Coppersmith, 1996a) which is analysed in some detail. It also briefly considers polynomial equations in more variables.

Chapter 5 looks at the factorisation equation $xy = N$ showing that it can also be treated in a similar "modular" way. The method is extended to allow for factoring over Gaussian integers, factoring of numbers with repeated factors, and analysing factors which lie in residue classes.

Chapter 6 considers Wiener-type attacks, i.e. exploiting the use of a small private exponent in RSA cryptography. It gives an overview of the problem, and then shows that if one has many public exponents all corresponding to small private exponents modulo N, then one can improve on Wiener's original attack considerably.

7.1 Results

The section highlights the original work contained within this thesis. This starts with Definition 2.2.3 which defines the concept of T-reliant addition chains. It is used to put the work described in (Brickell *et al.*, 1992) in to an addition chain framework. We subsequently show that this work can be used to produce relatively efficient addition chains for sets of integers.

Later in Chapter 2 we define the concept of deficient numbers (see Definition 2.4.1). We firstly show how to classify (Theorem 2.4.2) and produce (Theorem 2.4.3) such numbers, and then show that 1-deficient numbers may be used to approximately halve the time needed to detect torsion in group orders of known smoothness (see Algorithm 2.4.4). We also show that general n-deficient numbers may be useful with enough processors (see Section 2.4.2).

In Chapter 3 the first piece of seemingly original work[1] is Lemma 3.2.5 and the subsequent corollaries, in which it is shown how to deduce lower bounds for $\prod \lambda_i$ by examining the diagonal entries of a given triangular basis. This allows one to put definite bounds on the size of the vectors in an LLL-reduced basis as shown in equation 3.19, which in turn allows a slightly improved analysis on multivariate modular equations, as shown in section 4.7.

The next original concept in Chapter 3 is the notion of an effectively LLL-reduced basis. It is shown which of the classical results concerning LLL reduction rely on the "full" LLL conditions, and which just rely on the effective conditions. Also a slightly different variant of the LLL algorithm is described, where it is suggested that it may be better to do weak reduction at one block at the end, rather than all the way through the algorithm.

Another concept that relies only on an effectively LLL-reduced basis is the new result concerning LLL reduction on a given lattice basis or its dual. This is explained in Section 3.4.

[1] The result has a classical feel, but the author cannot find a previous reference to it. Certainly the use of this with LLL appears to be novel.

Section 3.5 then introduces the notion of unitary lattices, and extends the LLL algorithm to work over such structures. This cannot really be claimed as original work because of (Fieker & Pohst, 1996) and (Schiemann, 1998), though it was done independently.

In Chapter 4 the original work starts by describing the alternative method for solving univariate modular equations, given in Section 4.1. The connection between this and the existing method described in (Coppersmith, 1996b) is given in Section 4.4 and is shown to rely on the dual lattice results described in Section 3.4. We also (in Section 4.5) describe two small improvements to the general algorithm: removing one column of the matrix to be reduced, and a method to find slightly larger solutions for the same size of matrix.

The next piece of original work in Chapter 4 is the approach taken to break RSA when random padding is placed in more than one location. This work is described in Sections 4.7 and 4.8.4 and shows that there are provably weak ways to pad messages, even with many blocks. This is a partial answer to Open Problem 3 posed in (Boneh, 1999).

In Chapter 5 the original work starts by describing an alternative factoring algorithm to the one shown in (Coppersmith, 1996a). In fact the results achieved from the alternative algorithm are slightly better than a naïve use of Coppersmith's theorem.

It is then shown, in Section 5.3, that one can factor over the Gaussian integers, by using the LLL algorithm over unitary lattices described in Section 3.5. This is shown to have an impact on the moduli used in a cryptosystem by Vanstone and Zuccherato, essentially square rooting the time needed to factor them. This is an example of the fact that although Coppersmith has given a general algorithm for solving bivariate polynomial equations, the equations should still be treated with craft, and the immediate application of the algorithm may not always be the right approach.

Later in Chapter 5 we show how the lattice techniques can be used to factor numbers of the form $N = p^m q$. We give an $O(p^{1-m\alpha})$ for the factorisation of such moduli, where $p = N^\alpha$ i.e. $O(N^{1/8})$ even for the case $m = 2$. For large m this method becomes superior to the elliptic curve factorisation method, as shown in (Boneh *et al.*, 1999).

Chapter 5 ends by studying the problem of divisors in residue classes. The whole approach to this problem is original, but the results are a little disappointing. The main new result is the bound of $c(\alpha) = O((\alpha - 1/4)^{-3/2})$, but as shown in (Coppersmith *et al.*, 1998) Lenstra's analysis can also be extended to give this bound as well.

Finally Chapter 6 explores the interesting, if not a little theoretical problem, of breaking

RSA when one has many encrypting exponents e_i each with relatively small decrypting exponents d_i modulo a common modulus N. The lattice based solution is interesting and the general approach may be useful in other circumstances. It is shown that when one has very many e_i, $1 \leq i \leq m$, the d_i can be as large as $N^{1-\varepsilon}$. However the complexity of the algorithm is exponential in m and is certainly infeasible for $m \geq 10$ with present lattice reduction techniques.

7.2 Open problems

Having given a overview of the thesis, and highlighted the original contributions made by it, it remains to discuss some of the related problems that are still open. Chronologically, the first problem encountered is to better understand LLL over algebraic number fields (see Section 3.6), i.e. to work out the complexity of the reduction algorithm over given algebraic number fields, and to write efficient implementations of these. It would be nice to find more instances[2] of where the use such algorithms is helpful in cryptography.

In Section 4.7 we briefly discuss finding small solutions to multivariate Diophantine equations, and one particular method, using algebraic numbers, is shown for a class of equations arising from an application in RSA. Although, as shown in (Manders & Adleman, 1978) finding small solutions to even bivariate modular equations is NP-hard, it is an interesting problem to identify which instances of multivariate Diophantine equation are easier to solve than others.

The analysis of divisors in residue classes given in Section 5.5 is definitely an area for further work. The first goal is probably to align the results with (or possibly improve on) those given in (Lenstra, 1984). The ultimate goal is to join the upper and lower bounds described in Section 5.5.4, and thus produce exact upper bounds on the number of divisors in residue classes for given $\alpha > 1/4$.

Chapter 6 poses quite a few unanswered questions. On the problem of Wiener's attack it would be fascinating to increase the bound up to $N^{1/2-\varepsilon}$. If this could be done then perhaps the extended technique with many e_i could also be improved in a similar way. Conversely if one were to assume this, then it could possibly restrict the search for a solution to extending Wiener's original attack.

Further it would be rather nice to develop lattice techniques in which it can be proved (or at least good probabilities given) that techniques such as ours will definitely find the

[2] See Section 5.3 for one example.

desired vectors. It would also be nice to find other equations for which this approach (which is relatively general) can also be applied to.

Finally the problem of when the Chinese Remainder Theorem variant of RSA decryption is being used with small d_p and d_q seems very hard to attack. Perhaps the problem of only knowing that d_p and d_q are small could be shown to be NP-complete, or at least reduced to another supposedly hard poblem.

A final general point is that almost all the lattices in this thesis have quite discernible structure. It might possibly be the case that there exist more efficient ways to reduce such lattices, rather than resorting to straightforward implementations of LLL or its extensions.

Appendix A

Notation

Symbol	Meaning
$\mathbb{Z}, \mathbb{R}, \mathbb{C}$	The sets of integers, real numbers and complex numbers respectively
$\mathcal{G}$	The set of Gaussian integers
$\lvert a\rvert$	The absolute value of a real number a, or the length of a complex number a
$\lVert v\rVert$	the Euclidean norm of the vector v, i.e. $\left(\sum_{i=1}^{n} v_i^2\right)^{1/2}$
$\lVert v\rVert_1$	the Sum norm of the vector v, i.e. $\sum_{i=1}^{n} \lvert v_i\rvert$
$\lfloor x\rfloor$	The largest integer $\leq x$ when $x \in \mathbb{R}$
$\lceil x\rceil$	The smallest integer $\geq x$ when $x \in \mathbb{R}$
$\lfloor x\rceil$	The nearest integer to $x \in \mathbb{R}$
$GL_n(R)$	The set of invertible matrices over the ring R
$\mathcal{P}_B$	The set of primes $\leq B$
$\binom{n}{r}$	The number of ways of picking r objects from n
$f^{(i)}(x)$	The application of the function f, i times
$\lambda(N)$	The Carmichael lambda function (see page 6)
$\mu_{i,j}$	The coefficients resulting from the Gram-Schmidt orthogonalisation procedure (see page 38)

References

Aho, A., Hopcroft, J., & Ullman, J. 1974. *The design and analysis of computer algorithms*. Addison-Wesley, Reading, Mass.

Ajtai, M. 1998a. The shortest vector problem in L_2 is NP-hard for randomized reductions. *In: Proc. of STOC '98*.

Ajtai, M. 1998b. Worst-Case Complexity, Average-Case Complexity and Lattice Problems. *Pages 421–428 of: Proc. International Congress of Mathematicians*, vol. III.

Bach, Eric, & Peralta, René. 1996. Asymptotic semismoothness probabilities. *Mathematics of Computation*, **65**(216), 1701–1715.

Bleichenbacher, D. 1996. *Efficiency and security of cryptosystems based on number theory*. Ph.D. thesis, Swiss Federal Institute of Technology Zürich.

Bleichenbacher, D. 1998. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS# 1. *Pages 1–12 of: CRYPTO '98*. LNCS, vol. 1462. Springer.

Blömer, J., & Seifert, J.-P. 1999. On the Complexity of Computing Short Linearly Independent Vectors and Short Bases in a Lattice. *In: Proc. 31st Symposium on Theory of Computing*. To appear.

Boneh, D. 1999. Twenty years of attacks on RSA. *Notices of the AMS*, **46**, 203–213.

Boneh, D., & Durfee, G. 1999. New results on the cryptanalysis of low exponent RSA. *In: Proc. of EUROCRYPT '99*. To appear.

Boneh, D., & Venkatesan, R. 1998. Breaking RSA nay not be equivalent to factoring. *Pages 59–71 of: Eurocrypt '98*. LNCS, vol. 1403. Springer.

Boneh, D., Durfee, G., & Frankel, Y. 1998. An attack on RSA given a fraction of the private key bits. *In: Asiacrypt '98.*

Boneh, D., Durfee, G., & Howgrave-Graham, N. 1999. Factoring $N = p^r q$ for Large r. *In: CRYPTO '99.* To appear.

Brauer, A. 1939. On addition chains. *Bull. Amer. Math. Soc.*, **45**, 736–739.

Brickell, E., Gordon, D., McCurley, K., & Wilson, D. 1992. Fast exponentiation with precomputation. *Pages 200–207 of: EUROCRYPT '92.* LNCS, vol. 658. Springer.

Brlek, S., Castéran, P., & Strandh, R. 1991. On addition schemes. *Pages 379–393 of: TAPSOFT 1991.* LNCS, vol. 494.

Burrows, M., Abadi, M., & Needham, R. 1990. A logic of authentication. *ACM Trans. Computer Systems*, **8**, 18–36.

Cassels, J. W. S. 1971. *An introduction to the geometry of numbers.* Springer.

Cohen, H. 1991. *A course in computational algebraic number theory.* Springer-Verlag.

Conway, J., & Sloane, N. 1988. *Sphere packings, lattices and groups.* Grundlehren der math Wiss, no. 290. Springer, NY.

Coppersmith, D. 1996a. Finding a small root of a bivariate integer equation; factoring with high bits known. *In: Proceedings of Eurocrypt 96.*

Coppersmith, D. 1996b. Finding a small root of a univariate modular equation. *In: Proceedings of Eurocrypt 96.*

Coppersmith, D. 1998. Specialised integer factorisation. *In: Eurocrypt '98.*

Coppersmith, D., & Howgrave-Graham, N. A. 1999. *Using algebraic numbers to recover low-exponent RSA messages with padding in more than one location.* To appear.

Coppersmith, D., Franklin, M., Patarin, J., & Reiter, M. 1996. Low-Exponent RSA with Related Messages. *In: Proceedings of Eurocrypt 96.*

Coppersmith, D., Howgrave-Graham, N., & Nagaraj, S. V. 1998. *Divisors in residue classes – constructively.* To be submitted.

Coupé, C., Nguyen, P., & Stern, J. 1999. The effectiveness of lattice attacks against low-exponent RSA. *In: PKC '99.* To appear.

Davida, G. 1982. *Chosen signature cryptanalysis of the RSA (MIT) public key cryptosystem.* Tech. rept. TR-CS-82-2. Dept. of Electrical Engineering and Computer Science, University of Wisconsin, Milwaukee, USA.

de Bruijn, N. G. 1951. On the number of positive integers $\leq x$ and free of prime factors $> y$. *Indag. Math.*, **13**, 50–60. MR **13:**724e.

DeLaurentis, J. M. 1984. A further weakness in the common modulus protocol for the RSA cryptoalgorithm. *Cryptologia*, **8**, 253–259.

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

Downey, P., Leong, B., & Sethi, R. 1981. Computing sequences with addition chains. *SIAM J. Comput.*, **10**(3), 638–646.

Fieker, C., & Pohst, M. E. 1996. On lattices over Number Fields. *In: Proc. of Algorithmic Number Theory Sym. II.* LNCS, no. 1122. Springer.

Gauss, C. F. 1801. *Disquisitiones arithmeticae.* Leipzig.

Guo, C. R. 1996. An application of diophantine approximation in computer security. *Math. Comp.* To appear.

Hardy, G. H., & Wright, E.M. 1979. *An introduction to the theory of numbers.* 5 edn. Oxford University Press.

Hastad, J. 1988. Solving simultaneous modular equations of low degree. *SIAM J. of Computing*, **17**(2), 336–341.

Hermite, C. 1850. Extraits de lettres de M. Hermite à M. Jacobi sur différents objets de la théorie des nombres, deuxième lettre. *J. Reine Angew. Math*, **40**, 279–290.

Hildebrand, A. 1986. On the number of positive integers $\leq x$ and free of prime factors $> y$. *J. Number Theory*, **22**(3), 289–307.

Hildebrand, A., & Tenenbaum, G. 1986. On integers free of large prime factors. *Trans. Amer. Math. Soc.*, **296**(1), 265–290.

Howgrave-Graham, N. A. 1997. Finding small solutions of univariate modular equations revisited. *Pages 131–142 of: Cryptography and Coding.* LNCS, vol. 1355. Springer-Verlag.

Joux, A. 1993. *La réduction de réseaux en cryptographie.* Ph.D. thesis, Ecole polytechnique, Paris.

Joux, A., & Stern, J. 1998. Lattice reduction: a toolbox for the cryptanalyst. *J. Cryptology*, **11**(3), 161–185.

Joye, M. 1997. *Security analysis of RSA-type cryptosystems.* Ph.D. thesis, Univ. catholique de Louvain.

Jutla, C. 1998. On finding small solutions of modular multivariate polynomial equations. *Pages 158–170 of: Proc. of Eurocrypt '98.* Springer.

Kahn, D. 1967. *The Codebreakers.* Macmillan Publishing Company, New York.

Kannan, R. 1983. Improved algorithms for integer programming and related lattice problems. *Pages 193–206 of: Proc. 15th Symp. Theory of Comp.*

Knuth, D.E. 1981. *The Art of Computer Programming, vol. 2, Seminumerical Algorithms.* 2nd edn. Reading, Mass.: Addison-Wesley.

Kocher, P. 1996. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. *Pages 104–113 of: CRYPTO '96.* LNCS, vol. 1109. Springer.

Korkine, A., & Zolotarev, G. 1873. Sur les formes quadratiques. *Math. Ann.*, **6**, 336–389.

Lagrange, L. 1773. Recherches d'arithmétique. *Nouv. Mém. Acad. Sci., Paris.*

Lenstra, A. K., Lenstra, H. W., & Lovász, L. 1982. Factoring polynomials with integer coefficients. *Mathematische Annalen*, **261**, 513–534.

Lenstra, H. W. 1984. Divisors in residue classes. *Mathematics of Computation*, **42**(165), 331–340.

Lenstra, H. W. 1987. Factoring integers with elliptic curves. *Annals of Mathematics*, **126**, 649–673.

Manders, K. L., & Adleman, L. 1978. NP-Complete Decision Problems for Binary Quadratics. *J. Of Computer and System Sciences*, **16**, 168–184.

Matula, D.W. 1982. Basic digit sets for radix representation. *J. of the ACM*, **29**, 1131–1143.

McKee, J., & Pinch, R. 1998. On a cryptosystem of Vanstone and Zuccherato. *IEEE Transactions on Information Theory.* To appear.

McKee, J.M. 1990. *Distribution of the number of points on elliptic curves over a fixed finite prime field.* Ph.D. thesis, University of Cambridge.

Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. 1996. *Handbook of Applied Cryptography.* CRC Press.

Moore, J. H. 1992. Protocol failures in cryptosystems. *Contemporary Cryptology.*

Moree, Pieter. 1993. *Psixyology and Diophantine equations.* Ph.D. thesis, University of Leiden.

Okamoto, T., & Uchiyama, S. 1998. A new public-key cryptosystem as secure as factoring. *Pages 308–318 of: Proceedings of Eurocrypt 98.*

Pinch, R. 1997. *Mathematics for Cryptography.* Lecture notes for the Univ. of Cambridge.

Rivest, R. L., Shamir, A., & Adleman, L.M. 1978. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, **21**(2), 120–126.

RSA140. 1999. *Factorizaion of RSA-140 with the Number Field Sieve.* `sci.crypt.research`. 5th Feb.

Schiemann, A. 1998. Classification of hermitian forms with the neighbour method. *J. Symbolic Computation*, **26**, 487–508.

Schneier, B. 1996. *Applied cryptography : protocols, algorithms and source code in C.* 2nd edn. New York ; Chichester : Wiley.

Schnorr, C-P. 1987. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical computer science*, **53**, 201–224.

Schnorr, C-P. 1988. A more efficient algorithm for lattice basis reduction. *J. Algorithms*, 47–62.

Schnorr, C-P., & Euchner, M. 1991. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *In: Proc. of the FCT.* LNCS. Springer-Verlag, Berlin.

Shoup, V. 1995. *Number Theory Library (NTL).* `http://www.skoup.net`.

Silverman, J.H. 1986. *The Arithmetic of Elliptic Curves.* Springer-Verlag.

Simmons, G. J. 1983. A 'weak' privacy protocol using the RSA cryptalgorithm. *Cryptologia*, **7**, 180–182.

Stoll, R. R., & Wong, E. T. 1968. *Linear Algebra.* Academic Press.

Takagi, T. 1998. Fast RSA-type cryptosystem modulo $p^k q$. *Pages 318–326 of: Proceedings of CRYPTO 98.*

Tenenbaum, G. 1995. *Introduction to analytic and probabalistic number theory.* Cambridge studies in advanced mathematics, vol. 46. Camridge University Press.

Vallée, B., Girault, M., & Toffin, P. 1988. How to guess l'th roots modulo n by reducing lattice bases. *Pages 427–442 of: Proceeding of AAECC-6.* LNCS, vol. 357. Springer.

Vanstone, S. A., & Zuccherato, R. J. 1997. Elliptic curve cryptosystems using curves of smooth order over the ring Z_n. *IEEE Transactions of Information Theory*, **43**(4), 1231–1237.

Wiener, M. 1990. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory*, **36**(3), 553–558.

Yao, A.C. 1976. On the evaluation of powers. *SIAM J. Comput.*, **5**(1), 100–103.