ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΛΟΓΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΜΩΝ (CORELAB)

Πλέγματα και Εφαρμογές στο Κρυπτοσύστημα RSA

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Πέτρου Α. Μωλ

Επιβλέπων: Ευστάθιος Κ. Ζάχος
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2006

Εθνικο Μετσοβιο Πολυτεχνειο
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Πλέγματα και Εφαρμογές στο Κρυπτοσύστημα RSA

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Πέτρου Α. Μωλ

Επιβλέπων: Ευστάθιος Κ. Ζάχος
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 17η Ιουλίου 2006

....	....	....
Ε. Ζάχος	Τ. Σελλής	Φ. Αφράτη
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγήτρια Ε.Μ.Π.

Αθήνα, Ιούλιος 2006.

....
Πέτρος Α. Μωλ
Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι η μελέτη αφενός της μαθηματικής θεωρίας των πλεγμάτων και αφετέρου των εφαρμογών αυτής της θεωρίας στο κρυπτοσύστημα RSA. Στη βιβλιογραφία, ο όρος "πλέγμα" χρησιμοποιείται για την περιγραφή δύο αρκετά διαφορετικών μαθηματικών δομών. Η πρώτη δομή έχει να κάνει με μερικώς διατεταγμένα σύνολα ενώ η δεύτερη με περιοδικές διατάξεις σημείων στο χώρο. Στην εργασία αυτή ασχολούμαστε αποκλειστικά με τη δεύτερη δομή.

Ιστορικά, τα πλέγματα εμφανίστηκαν το 18^o αιώνα και μελετήθηκαν από μαθηματικούς όπως ο Langrange και ο Gauss. Το 19^o αιώνα, σημαντικά αποτελέσματα από τον Minkowski έδωσαν ώθηση στη χρήση της θεωρίας των πλεγμάτων στη θεωρία και στη γεωμετρία των αριθμών. Με την ανάπτυξη των υπολογιστών, η θεωρία των πλεγμάτων βρήκε εφαρμογές σε πολλά θεωρητικά πεδία, μεταξύ των οποίων η παραγοντοποίηση των ακέραιων πολυωνύμων, ο ακέραιος προγραμματισμός και η κρυπτογραφία δημοσίου κλειδιού. Ειδικά στην κρυπτογραφία δημοσίου κλειδιού, με την οποία και θα ασχοληθούμε, τα πλέγματα χρησιμοποιήθηκαν στην ανάπτυξη νέων κρυπτοσυστημάτων, στη θεμελίωση κρυπτογραφικών προτύπων καθώς και στην κρυπτανάλυση.

Πρωταρχικός στόχος κάθε κρυπτοσυστήματος είναι η εξασφάλιση της μετάδοσης της πληροφορίας από το νόμιμο αποστολέα στους νόμιμους αποδέκτες χωρίς να είναι εφικτή η άντληση μέρους της πληροφορίας από κάποια τρίτη , μη εξουσιοδοτημένη, οντότητα. Στα πλαίσια της εργασίας μελετάται αποκλειστικά το κρυπτοσύστημα RSA, ίσως ο πιο διαδεδομένος εκπρόσωπος των κρυπτοσυστημάτων δημοσίου κλειδιού. Από το 1977 οπότε και προτάθηκε από τους Rivest, Shamir και Adleman, το RSA χρησιμοποιείται κατά κόρον σε εφαρμογές όπου η ασφάλεια και η μυστικότητα είναι θεμελιώδους σημασίας, όπως στην ανταλλαγή μηνυμάτων μέσω ηλεκτρονικού ταχυδρομείου, στην ψηφιακή υπογραφή εγγράφων και στις πληρωμές με χρήση πιστωτικών καρτών. Τα 30 περίπου αυτά χρόνια της ύπαρξης του, το RSA έχει μελετηθεί εκτενώς τόσο ως προς τις διάφορες παραλλαγές του που έχουν προταθεί, όσο και ως προς την αντοχή του σε επιθέσεις.

Στο πρώτο κεφάλαιο δίνονται βασικοί ορισμοί και ιδιότητες των πλεγμάτων ενώ στο δεύτερο μελετάται η έννοια της αναγωγής βάσης ενός πλέγματος με έμφαση στην LLL αναγωγή και στον ομώνυμο αλγόριθμο. Στο τρίτο κεφάλαιο παρουσιάζεται η ενσωμάτωση της θεωρίας των πλεγμάτων στην επίλυση πολυωνυμικών εξισώσεων. Στο τέταρτο κεφάλαιο δίνεται συνοπτική παρουσίαση του κρυπτοσυστήματος RSA και παρουσιάζεται μία θετική εφαρμογή των πλεγμάτων σε αυτό. Τέλος, στο πέμπτο κεφάλαιο παρουσιάζονται οι σημαντικότερες σχετιζόμενες με πλέγματα επιθέσεις εναντίον του RSA και αναλύεται η αντοχή των διαφόρων παραμέτρων του κρυπτοσυτήματος στις επιθέσεις αυτές.

Λέξεις Κλειδιά: Πλέγμα, Αναγωγή Βάσης Πλέγματος, Αλγόριθμος LLL, Κρυπτοσύστημα RSA, Επιθέσεις με Πλέγματα.

Abstract

The purpose of this diploma thesis is the study both of the mathematical background on lattice theory and of the corresponding applications to the RSA Cryptosystem. In bibliography, there are two quite different mathematical structures that are usually called lattices. The first one has to do with partially ordered sets while the other has to do with regular arrangements of points in space. In this thesis we exclusively consider the second case.

Historically, lattices were investigated since the late 18th century by mathematicians such as Lagrange and Gauss. In the 19th century, important results due to Minkowski motivated the use of lattice theory in the theory and geometry of numbers. The evolution of computer science in the 20th century led to lattice applications in various theoretical areas such as factorization of integer polynomials, integer programming and Public-Key Cryptography. In the latter area, lattice theory has played a crucial role in the definition of new cryptosystems, in the study of cryptographic primitives and in cryptanalysis.

The main goal of a cryptosystem is to guarantee the exchange of information between the legitimate sender and the legitimate receivers, ensuring at the same time, that any unauthorized party is unable to recover part of the information. In the thesis in hand, we focus exclusively on RSA Cryptosystem, which is probably the most wide-spread Public-Key Cryptosystem. Since its publication, in 1977, RSA has been used in plenty of applications ranging from digital signatures to electronic credit-card payment systems. After almost 30 years of existence, RSA has been extensively analyzed for vulnerabilities by many researchers.

In the first chapter we give some basic background on lattices while, in the second, we introduce the notion of lattice basis reduction with emphasis to LLL reduction and the corresponding algorithm. The third chapter describes the use of lattices in finding small roots to polynomial equations. In the fourth chapter, we present RSA and a positive lattice related application to it. Finally, in the fifth chapter, we present an overview of the most representative lattice-based attacks mounted against RSA since its publication.

Key Words: Lattice, Lattice Reduction, LLL Algorithm, RSA Cryptosystem, lattice Attacks

Contents

List of Algorithms

Chapter 1

Introduction to Lattices

In this chapter we give the mathematical background on lattices. The chapter mainly includes definitions about lattices, some very useful lattice properties and some necessary theorems that will allow the reader to follow the material presented in the next chapters.

1.1 Linear Algebra Preliminaries

Before we proceed to a formal definition of lattices and give some relevant important theorems, we will summarize some notation necessary for the rest of the analysis and some properties of mathematical structures we will use throughout this thesis.

1.1.1 Notation

We summarize below some of the notation used:

- Let S be a set. Then $M_{m,n}$ will denote the set of all $m \times n$ matrices with entries from S.
- If M denotes a matrix, then M^T will denote the transpose matrix of M.
- $\lceil a \rceil$, $\lfloor a \rfloor$ will denote the ceiling (that is, the smallest integer not smaller than a) and the floor (that is, the biggest integer not bigger than a) of integer a respectively, while $\lceil a \rfloor$ will denote the integer closest to a, namely $\lceil a \rfloor \equiv \lceil a - 0.5 \rceil$.

1.1.2 Scalar Product

Definition 1.1.1 (**Scalar Product**)
Let $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ be a mapping with the following properties :
$\forall u, v, w \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$:

1. $\langle u+w, v\rangle = \langle u, v\rangle + \langle w, v\rangle$
   $\langle \lambda u, v\rangle = \lambda\langle u, v\rangle$
   $\langle u, v+w\rangle = \langle u, v\rangle + \langle u, w\rangle$
   $\langle u, \lambda v\rangle = \lambda\langle u, v\rangle$
2. $\langle u, v\rangle = \langle v, u\rangle$ and
3. $\langle u, u\rangle > 0$ for $u \neq 0$.

We call such a mapping **scalar product**. Properties 1, 2 and 3 imply that the scalar product is bilinear , symmetric and positive definite respectively.

Definition 1.1.2 (**Standard Scalar Product**)
The **standard scalar product** is defined as:

$$\langle (u_1, u_2, ..., u_n)^T, (v_1, v_2, ..., v_n)^T\rangle := \sum_{i=1}^{n} u_i v_i \tag{1.1}$$

and will be the scalar product used in most of the cases in this thesis.

1.1.3 Norm

Let $\mathbb{F}$ be any field. The vector space $\mathbb{F}^n$ is the set of all n-tuples $\vec{x} = (x_1, x_2, ..., x_n)$ where $x_i \in \mathbb{F}$ are field elements. We are mainly interested in vector spaces over the reals or over the rationals ($\mathbb{F} = \mathbb{R}\, or\, \mathbb{F} = \mathbb{Q}$).

Definition 1.1.3 (**Norm**)
Let $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ be a mapping such that $\forall\, u, v \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$:

1. $\|\lambda u\| = |\lambda| \cdot \|u\|$ (positive homogeneous)
2. $\|u+v\| \leq \|u\| + \|v\|$ (triangle inequality)
3. $\|u\| > 0$ for $u \neq 0$ (positive definiteness)

We call such a mapping **norm** (or length) of vector $u = (u_1, u_2, ..., u_n)$.

In general we define l_p norm as:

$$l_p = \|(u_1, u_2, ..., u_n)^T\|_p := (\sum_{i=1}^{n} |u_i|^p)^{\frac{1}{p}}. \tag{1.2}$$

More specifically for $p = 1, p = 2$ and $p = \infty$ and for the standard scalar product notation we get:

$$\begin{array}{ll} l_1 = \|(u_1, u_2, ..., u_n)^T\|_1 := \sum_{i=1}^{n} |u_i| & (l_1\, norm) \\ l_2 = \|(u_1, u_2, ..., u_n)^T\|_2 := \sqrt{\langle u, u\rangle} := (\sum_{i=1}^{n} u_i^2)^{1/2} & (l_2\, norm) \\ l_\infty = \|(u_1, u_2, ..., u_n)^T\|_\infty := \max_{i=1,2,...,n} |u_i| & (l_\infty norm) \end{array}$$

These are the norms that we will deal with in this thesis. When we use the symbol $\|u\|$ without an index, we will always imply l_2 (Euclidean Norm) unless otherwise stated.

Every norm $\|\cdot\|$ induces a distance function $d(\vec{x},\vec{y}) = \|\vec{x}-\vec{y}\|$. The distance function induced by the Euclidean norm is the usual Euclidean distance.

Finally we say that two vectors are orthogonal ($\vec{x}\perp\vec{y}$) if $\langle\vec{x},\vec{y}\rangle = 0$.

1.1.4 Orthogonal bases (Gram-Schmidt orthogonalization)

Gram-Schmidt orthogonalization is a fundamental procedure in linear algebra.It transforms any set of n linear independent vectors into a set of n orthogonal vectors by projecting each vector on the space orthogonal to the span of the previous vectors (or equivalently by removing from each vector the components that belong to the vector spanned by the previous vectors). Figure 1.1 illustrates the method for a 2-dimensional space.

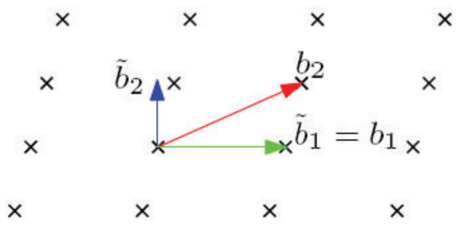

Figure 1.1: Gram-Schmidt orthogonalization

Definition 1.1.4 (**Gram-Schmidt Ortogonalization**)
Let $b_1, b_2, ..., b_n$ be a sequence of n linearly independent vectors.We define their **Gram-Schmidt orthogonalization** as the sequence $b_1^*, b_2^*, ..., b_n^*$ defined by:

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*,\ where\ \mu_{i,j} = \frac{\langle b_i, b_j^*\rangle}{\langle b_j^*, b_j^*\rangle}. \tag{1.3}$$

In other words, b_i^* is the component of b_i that is orthogonal to $b_1^*, ..., b_{i-1}^*$.

Below (algorithm 1) we give the algorithm that performs the Gram-Schmidt Orthogonalization.

Remark 1.1.5. From the above definition it is fairly easy to derive some useful properties of Gram-Schmidt orthogonalization.

(a) It is trivial to verify that $\langle b_i^*, b_j^*\rangle = 0$ for each $i \neq j$.

Algorithm 1: Gram-Schmidt Orthogonalization (GSO)

```
Input: Linearly Independent Vectors $b_1, b_2, ..., b_n \in \mathbb{R}^n$
Output: The Gram-Schmidt orthogonalization of $b_1, b_2, ..., b_n$ :
        $b_1^*, b_2^*, ..., b_n^*$ and $\mu_{i,j}$ for $1 \leq j < i \leq n$
begin
    $b_1^* \leftarrow b_1$;
    for $i \leftarrow 2$ to $n$ do
        $b_i^* \leftarrow b_i$;
        for $j \leftarrow 1$ to $i-1$ do
            $\mu_{i,j} \leftarrow \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}$;
            $b_i^* \leftarrow b_i^* - \mu_{i,j} b_j^*$ ;
        end
    end
    return $b_1^*, b_2^*, ..., b_n^*$ and $\mu_{i,j}$ for $1 \leq j < i \leq n$.
end
```

(b) $\forall 1 \leq i \leq n$, $span(b_1, b_2, ..., b_n) = span(b_1^*, b_2^*, ..., b_n^*)$. This is actually a very significant property that will be frequently used throughout the rest of the chapter.

(c) The order of the vectors $b_1, b_2, ..., b_n$ is important,that's why we consider them as a sequence and not as a set.

(d) If we rewrite the equation for the orthogonal vectors in the form

$$b_i = b_i^* + \sum_{j=1}^{i-1} \mu_{i,j} b_j^* \tag{1.4}$$

we can obtain a relation between the initial basis $B = [b_1, b_2, ..., b_n]$ and the orthogonal one $B^* = [b_1^*, b_2^*, ..., b_n^*]$ in the following matrix form:

$$\left[b_1, b_2, ..., b_n\right] = \left[b_1^*, b_2^*, ..., b_n^*\right] \cdot \left[\mu_{i,j}\right]_{1 \leq i,j \leq n}^T \tag{1.5}$$

or equivalently

$$\left[b_1, b_2, ..., b_n\right] = \left[b_1^*, b_2^*, ..., b_n^*\right] \cdot \begin{bmatrix} 1 & \mu_{2,1} & \mu_{3,1} & \cdots & \mu_{n,1} \\ 0 & 1 & \mu_{3,2} & \cdots & \mu_{n,2} \\ \vdots & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mu_{n,n-1} \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}. \tag{1.6}$$

1.1.5 Useful Inequalities

We finish this section by presenting some useful inequalities in linear algebra.

1. **Cauchy-Schwarz :** Let V be a vector space where a standard scalar product has been defined. Let $u, v \in V$ be two vectors. Then

$$|\langle u, v\rangle|^2 \leq \langle u, u\rangle \cdot \langle v, v\rangle.$$

The equality holds when u, v are linearly dependent.

Proof. The proof of the inequality is trivial if $v = 0$. Consider now the case where $v \neq 0$. Then $\langle v, v\rangle > 0$. In addition $\langle x, x\rangle \geq 0$ for any vector $x \in V$. Thus $\langle u - \lambda v, u - \lambda v\rangle \geq 0$ for all $\lambda \in \mathbb{R}$. This gives

$$0 \leq \langle u - \lambda v, u - \lambda v\rangle = \langle u, u\rangle + \lambda^2 \langle v, v\rangle - 2\lambda\langle u, v\rangle$$

where we have used the properties of the scalar product. If we now choose

$$\lambda = \frac{\langle u, v\rangle}{\langle v, v\rangle}$$

the above inequality becomes

$$0 \leq \langle u, u\rangle - \frac{|\langle u, v\rangle|^2}{\langle v, v\rangle} \Rightarrow |\langle u, v\rangle|^2 \leq \langle u, u\rangle \cdot \langle v, v\rangle.$$

The equality holds if $u - \lambda v = 0$ that is when u, v are linearly dependent. □

If we consider the standard scalar product and the euclidean norm (where $\langle x, x\rangle = \|x\|^2$) the Cauchy-Schwarz inequality takes the following interesting form

$$|\langle u, v\rangle| \leq \|u\| \cdot \|v\|.$$

2. **Norm Inequalities :** From the previous definitions of l_1, l_2, l_∞ we get the following obvious inequalities:

$$\|u\|_2 \leq \|u\|_1 \leq \sqrt{n} \cdot \|u\|_2 \tag{I1}$$

$$\|u\|_\infty \leq \|u\|_2 \leq \sqrt{n} \cdot \|u\|_\infty \tag{I2}$$

$$\|u\|_\infty \leq \|u\|_1 \leq n \cdot \|u\|_2 \tag{I3}$$

These inequalities can easily be derived directly from the definition of the norms, except from the second part of inequality $(I1)$.For that one we need Cauchy-Schwarz inequaility with $u = (1, 1, ..., 1)^T$ and $v = (|u_1|, |u_2|, ..., |u_n|)^T$.

3. **Hadamard Inequality :** Let $b_1, b_2, ..., b_n \in \mathbb{R}^n$ be the column vectors of the matrix $B \in M_{n,n}(\mathbb{R})$. Then by Hadamard's inequality we have:

$$|detB| \leq \prod_{i=1}^{n} \|b_i\|_2. \tag{1.7}$$

The equality holds when the vectors $b_1, b_2, ..., b_n$ are orthogonal.

Proof. Let B^* be the corresponding Gram-Schmidt basis of B. We know that $det(B)^2 = det(B^T B)$. Let $\bar{\mu}$ denote the matrix with entries $[\mu_{i,j}]$ where $\mu_{i,j}$ are the Gram-Schmidt coefficients. Then equation 1.5 says that

$$B = B^* \bar{\mu}^T.$$

Notice that $\bar{\mu}$ is lower triangular with determinant 1. Thus we have

$$det(B)^2 = det(B^T B) = det(\bar{\mu} \cdot (B^*)^T \cdot B^* \cdot \bar{\mu}^T) = det((B^*)^T \cdot B^*)$$

since $det(\bar{\mu}) = det(\bar{\mu}^T) = 1$. The matrix $(B^*)^T \cdot B^*$ is an $n \times n$ matrix with entries $\langle b_i^*, b_j^* \rangle$. Since $\langle b_i^*, b_j^* \rangle = 0$ for $i \neq j$ and $\langle b_i^*, b_i^* \rangle = \|b_i^*\|^2$, $(B^*)^T \cdot B^*$ is a matrix with only diagonal entries $\|b_i^*\|^2$. Hence, its determinant is equal to $(\prod_{i=1}^n \|b_i^*\|)^2$. The above observations leat to

$$|det(B)| = \sqrt{det(B^T B)} = \sqrt{det((B^*)^T \cdot B^*)} = \prod_{i=1}^{n} \|b_i^*\|.$$

In addition equation 1.4 along with the fact that b_i^*, b_j^* are pairwise orthogonal give that

$$\|b_i\|^2 = \|b_i^*\|^2 + \sum_{j=1}^{i-1} \mu_{i,j}^2 \|b_j^*\|^2 \geq \|b_i^*\|^2 \Rightarrow \|b_i\| \geq \|b_i^*\|.$$

Thus

$$|det(B)| = \prod_{i=1}^{n} \|b_i^*\| \leq \prod_{i=1}^{n} \|b_i\|$$

which completes the proof.Obviously the equality holds whenever the initial vectors b_i are pairwise orthogonal. In that case the corresponding Gram-Schmidt basis B^* is identical to B. □

1.2 Basic Definitions on Lattices

We start by giving a formal definition of a lattice.

Definition 1.2.1 (**Lattice, Basis , Rank , Dimension**)
Let $B = \{b_1, b_2, ..., b_n\}$ be a set of n *linearly independent vectors* in $\mathbb{R}^m$. The **lattice** generated by B is the set

$$\mathcal{L}(B) = \{\sum_{i=1}^{n} x_i \cdot \vec{b_i} \ : \ x_i \in \mathbb{Z}\}. \tag{1.8}$$

That is, the set of all *integer linear combinations* of the basis vectors.
The set B is called **basis** and we can compactly represent it as an $m \times n$ matrix each column of whose is a basis vector:

$$B = [b_1, b_2, ..., b_n].$$

The **rank** of the lattice is defined as $rank(\mathcal{L}) := n$ while its **dimension** is defined as $dim(\mathcal{L}) := m$.

Remark 1.2.2. In this thesis we will mainly consider full-rank lattices, that is lattices where $n = m$.

Remark 1.2.3. It is important to emphasize straight from the beginning the difference between a lattice and a vector space.Compare the definition given above to the vector space definition.

Definition 1.2.4 (**Vector Space**)
Let $B = \{b_1, b_2, ..., b_n\}$ be a set of n *linearly independent vectors* in $\mathbb{R}^m$. The **vector space** generated by B is the set

$$span(B) = \{\sum_{i=1}^{n} x_i \cdot \vec{b_i} \ : \ x_i \in \mathbb{R}\} = \{B \cdot \vec{x} \ : \ x \in \mathbb{R}^n\}.$$

That is, the set of **all** linear combinations of the basis vectors.

Apparently the difference lies to the coefficients x_i which are integers in the case of a lattice instead of reals in the case of a vector space.

Remark 1.2.5. The definition of $\mathcal{L}(B)$ makes sense even if the vectors b_i are not linearly independent. However, in that case, $\mathcal{L}(B)$ is not necessarily a lattice. That is, a (possibly) smaller set of linearly independent vectors B' such that $\mathcal{L}(B') = \mathcal{L}(B)$ **does not** necessarily exist. To see that consider the trivial case where $B = [1, a]$ with a being an irrational number. Then $\mathcal{L}(B) = \{x + ya : x, y \in \mathbb{Z}\}$. Clearly B is not a set of linearly independent vectors. In addition $\mathcal{L}(B)$ is not a lattice since there does not exist an l such that $\mathcal{L}(B) = l \cdot \mathbb{Z}$ because of the irrationality of a. In the rest of the thesis the symbol $\mathcal{L}(B)$ or simply $\mathcal{L}$ will imply a lattice unless otherwise mentioned.

We give now some trivial examples of two-dimesional lattices.

Example 1.2.6. Let us first consider a very simple example of a lattice in the 2-dimensional space. Let

$$b_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } b_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Apparently b_1, b_2 are linearly independent.The lattice generated by this two vectors is $\mathbb{Z}^2$ (see figure 1.2).

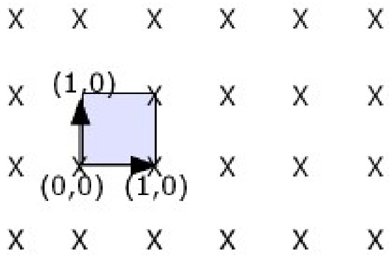

Figure 1.2: A basis of $\mathbb{Z}^2$

Remark 1.2.7. b_1, b_2 are not the only vectors that produce $\mathbb{Z}^2$. Consider, for instance, the pair of vectors

$$b_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \text{ and } b_2 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}.$$

It is obvious that this pair generates exactly the same lattice (see figure 1.3). Actually, each lattice has infinetely many bases.However, not each pair of linearly

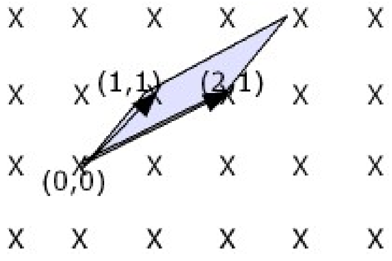

Figure 1.3: Another basis of $\mathbb{Z}^2$

independent vectors can produce a specific lattice. Consider for example the pair

$$b_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } b_2 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}.$$

b_1, b_2 are clearly linearly independent but they cannot generate $\mathbb{Z}^2$ (see figure 1.4).Indeed, they cannot produce the point (1,1), that is there is no integer pair

(x, y) such that

$$x \cdot \vec{b_1} + y \cdot \vec{b_2} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

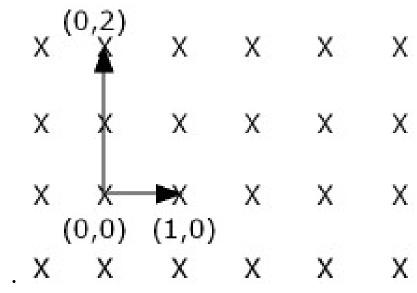

Figure 1.4: Basis that cannot produce $\mathbb{Z}^2$

The previous examples raise the question whether a given set of vectors forms a basis of a lattice or not. In the next few lines we will try to give an answer to that question.

Definition 1.2.8 (Fundamental Parallelepiped)
For any lattice basis B we define

$$\mathcal{P}(B) = \{Bx | x \in \mathbb{R}^n, \forall i : 0 \leq x_i < 1\}. \tag{1.9}$$

Remark 1.2.9. Note that $\mathcal{P}(B)$ is half-open. This implies that the translates $\mathcal{P}(B) + \vec{u}$ (where $\vec{u} \in \mathcal{L}(B)$) form a partition of the whole space $\mathbb{R}^n$.

Two examples of the fundamental parallelepiped are shown in figures 1.2 and 1.3. It is obvious from those two examples that the fundamental parallelepiped greatly depends on the specific basis of the lattice. The following theorem gives us a criterion to test whether a given set of n linearly independent vectors $b_1, b_2, ..., b_n$ form a basis of a given lattice $\mathcal{L}$ or not.

Theorem 1.2.10
Let $\mathcal{L}$ be a lattice of rank n and $b_1, b_2, ..., b_n \in \mathcal{L}$ be n linearly independent lattice vectors. Then $b_1, b_2, ..., b_n$ form a basis of $\mathcal{L}$ if and only if $\mathcal{P}(b_1, b_2, ..., b_n) \cap \mathcal{L} = \{0\}$.

Proof. ($\Rightarrow$) Let $b_1, b_2, ..., b_n$ form a basis of $\mathcal{L}$. Then,by definition, $\mathcal{L}$ is the set of all their linear integer combinations. In addition $\mathcal{P}(b_1, b_2, ..., b_n)$ is, again by definition, the set of linear combinations of $b_1, b_2, ..., b_n$ with coeficients in $[0, 1)$.Since the right side of the interval is open, the only integer combination that belongs in $\mathcal{P}$ is the one where $x_i = 0 \, \forall i$ and therefore the intersection of the two sets is clearly $\{0\}$.
($\Leftarrow$) Assume now that $\mathcal{P}(b_1, b_2, ..., b_n) \cap \mathcal{L} = \{0\}$. Since $b_1, b_2, ..., b_n$ are linearly

independent, we can express any lattice vector x $\in \mathcal{L}$ as $\sum y_i b_i$ for some $y_i \in \mathbb{R}$. Let $x' = \sum (y_i - \lfloor y_i \rfloor) b_i$. Then $x' \in \mathcal{L}$ since by definition a lattice is closed under addition and obviously $x' \in \mathcal{P}(b_1, b_2, ..., b_n)$ as $(y_i - \lfloor y_i \rfloor) \in [0, 1)$. Thus, $x' = 0$ by our assumption which along with the linear independency of $b_1, b_2, ..., b_n$ gives that $y_i = \lfloor y_i \rfloor\ \forall y_i \in \mathbb{R}$ and therefore the arbitrary $x \in \mathcal{L}$ can be expressed as an integer combination of $b_1, b_2, ..., b_n$ which implies that $b_1, b_2, ..., b_n$ form a basis of $\mathcal{L}$. □

Remark 1.2.11. A restatement of the above theorem would be "For all $\vec{x} \in \mathbb{R}^n$, there exists a unique lattice point $\vec{u} \in \mathcal{L}(B)$ such that $\vec{x} \in (\mathcal{P}(B) + \vec{u})$".

Despite the relatively simple condition of the above theorem, we cannot apply it in a straightforward fashion. Instead, what we can actually do, is verify whether two (different) sets of linear independent vectors can produce the same lattice or , expressed in another way, whether two bases B_1, B_2 are equivalent.We first give some definitions.

Definition 1.2.12 (**Unimodular Matrix**)
A matrix $U \in \mathbb{Z}^{n \times n}$ is called **unimodular** if $detU = \pm 1$.We will use $GL_n(\mathbb{Z})$ to denote the group of integer $n \times n$ matrices with determinant ± 1.

$$GL_n(\mathbb{Z}) := \{U \in M_{n,n}(\mathbb{Z}) | detU = \pm 1\}. \tag{1.10}$$

Theorem 1.2.13
$GL_n(\mathbb{Z})$ is a group under matrix multiplication.

Proof. First, if $U_1, U_2 \in GL_n(\mathbb{Z})$, then $U_1 \cdot U_2 \in \mathbb{Z}^{n \times n}$ and $det(U_1 \cdot U_2) = det(U_1) \cdot det(U_2) = 1$ which means that GL_n is closed under matrix multiplication. In addition the identity matrix is apparently unimodular. Moreover let $U \in GL_n(\mathbb{Z})$. Then $det(U)det(U^{-1}) = 1$ implies that $det(U^{-1}) = \pm 1$ which, along with the Cramer's rule, gives that the (i, j) entry in the U^{-1} matrix is:

$$\frac{(-1)^{i+j} \cdot det(T_{ij})}{det(U)} = \pm U_{ij}$$

where T_{ij} denotes the algebraic complement of each element of U (that is U with the ith and jth column deleted). Thus, every entry of U^{-1} is an integer (since T_{ij} is an integer matrix and therefore its determinant is an integer) which gives that $U^{-1} \in GL_n(\mathbb{Z})$. Finally the associativity holds since matrix multiplication is associative. □

Consider now the following elementary column operations:

1. Exchange of two columns.
2. Multiplication of a column by -1.
3. Addition of an integer multiple of one column to another.

It is not difficult to show that each one of the above elementary operations can be performed on a matrix by right-multiplication (or left-multiplication) with an appropriately chosen unimodular matrix. We will show how this can be performed in the case of $\Pi_{2,2}$ matrices. The extension to $\Pi_{n,n}$ matrices is trivial.
Let

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

Then it is not difficult to verify that right multiplication of A with

$$E_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad E_2 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \text{ and } E_3 = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix},$$

performs respectively the 3 above elementary operations.

We now turn our attention to the condition two bases should satisfy in order to be equivalent. Such a condition is given in the follwoing very important theorem.

Theorem 1.2.14 (**Bases Equivalence**)
Two bases $B_1, B_2 \in \mathbb{R}^{m\times n}$ are **equivalent** *if and only if $B_2 = B_1U$ for some unimodular matrix U.*

Proof. ($\Rightarrow$) Assume first that B_1, B_2 are equivalent, that is they produce the same lattice. Then, for each of the n columns of b_i of B_2, $b_i \in \mathcal{L}(B_1)$. This means that each b_i of B_2 can be expressed as a linear integer combination of the column vectors of B_1 and therefore there exists $U \in \mathbb{Z}^{n\times n}$ such that $B_2 = B_1U$. Similarly, there exists $V \in \mathbb{Z}^{n\times n}$ such that $B_1 = B_2V$ which implies that $B_2 = B_1U = B_2VU$. Hence

$$\begin{aligned} B_2^TB_2 = (VU)^TB_2^TB_2(VU) &\Rightarrow det(B_2^TB_2) = (det(VU))^2det(B_2^TB_2) \\ &\Rightarrow det(U)det(V) = \pm 1 \end{aligned}$$

Since U, V are both integer matrices this means that $det(U) = \pm 1$.

($\Leftarrow$) The hypothesis that $B_2 = B_1U$ for some unimodular matrix U means that each column of B_2 is contained in $\mathcal{L}(B_1)$ which impies that $\mathcal{L}(B_2) \subseteq \mathcal{L}(B_1)$. In addition,$B_1 = B_2U^{-1}$. But we have shown that U^{-1} is unimodular with integer entries ($GL_n(\mathbb{Z})$ is a group under matrix multiplication).So $\mathcal{L}(B_1) \subseteq \mathcal{L}(B_2)$ which finally gives $\mathcal{L}(B_2) = \mathcal{L}(B_1)$. □

Remark 1.2.15. As an immediate corollary, we obtain that B is a basis of $\mathbb{Z}^n$ if and only if it is unimodular.

Remark 1.2.16. Since elementary column operations can be performed by right multiplication of the basis matrix by a unimodular matrix (we showed that earlier), if we modify basis $B = [b_1, b_2, ..., b_n]$ by:

1. Reordering the columns.

2. Multiplying any number of columns by -1.
3. Adding integer multiples of some columns to others.

then the resulting matrix will still be a basis for the same lattice.

Interestingly, the inverse is true too.

Theorem 1.2.17
Two bases are equivalent if and only if one can be obtained from the other by the following elementary operations on the columns:

1. $b_i \leftarrow b_i + kb_j$ *for some* $k \in \mathbb{Z}$
2. $b_i \leftrightarrow b_j$
3. $b_i \leftarrow -b_i$

1.3 Determinant

We now give the definition of a very important characteristic of a lattice, namely its determinant.

Definition 1.3.1 (**Determinant**)
The **determinant** $det(\mathcal{L})$ of a lattice $\mathcal{L}(b_1, b_2, ..., b_n) \subseteq \mathbb{R}^m$ is generally defined as :

$$det(\mathcal{L}) = (det[\langle b_i, b_j \rangle]_{1 \leq i,j \leq n})^{\frac{1}{2}}. \tag{1.11}$$

Remark 1.3.2. This is the general definition. In this general case we form an $n \times n$ matrix D with $D_{ij} = \langle b_i, b_j \rangle$, where $\langle b_i, b_j \rangle$ denotes the scalar product of vectors b_i, b_j in the general case (see subsection 1.1.2). If we restrict our definition to the case of *standard scalar product* then the determinant is defined in a more compact form as follows:

$$detL = \sqrt{det(B^T B)}.$$

In this thesis we will always imply the standard scalar product when talking about the determinant. More interestingly, in the case where $m = n$ (full rank lattices), B is a square matrix and $det(\mathcal{L}) = |detB|$. The latter definition will be used almost exclusively throughout this thesis.

An alternative definition of the determinant in the standard scalar product case is the following:

Definition 1.3.3
The determinant of the a lattice $\mathcal{L}$ is defined as the n-dimensional volume of the fundamental parallelepided associated to B:

$$det(\mathcal{L}(B)) = vol(\mathcal{P}(B)).$$

The following theorem shows that the determinant of a lattice is well-defined, that is, it is independent of the choice of the basis B. We can therefore write either $det(B)$ or $det(\mathcal{L})$ and mean the exact same thing.

Theorem 1.3.4
The determinant of a lattice is independent of the choice of the basis $b_1, b_2, ..., b_n \in \mathbb{R}^m$.

Proof. Let B_1, B_2 be two bases of lattice $\mathcal{L}$. Theorem 1.2.14 states that there is a unimodular matrix U such that $B_2 = B_1 U$.Thus

$$det\mathcal{L} = \sqrt{det(B_2^T B_2)} = \sqrt{det(U^T B_1^T B_1 U)} = \sqrt{det(B_1^T B_1)}$$

which completes the proof. □

As an immediate result of the above theorem and theorem 1.2.14 we get the following corollary for the standard notion of scalar product.

Corollary 1.3.5. If two bases $B_1, B_2 \in \mathbb{R}^{m \times n}$ are **equivalent** then $|det(B_2) = |det(B_1)|$. The opposite is not necessarily true.

1.4 Successive Minima

After presenting most of the important properties of a lattice, we proceed now by defining and studying another very important characteristic of a lattice, namely its shortest vector. By the definition of a lattice, it is obvious that $\vec{0} \in \mathcal{L}$ for every lattice (we just have to consider a linear combination of b_is with the null vector). Thus $\vec{0}$ is always excluded from our discussion.

1.4.1 Definitions

Definition 1.4.1 (Shortest Vector)
Let $\| \cdot \|$ be an arbitrary norm. The **shortest vector** of the lattice is defined as the non-zero vector $\vec{u} \in \mathcal{L}$ such that its norm is minimal. Expressed in a different way, the shortest vector is a vector $\vec{u} \in \mathcal{L}(B) \backslash \{\vec{0}\}$ such that $\|\vec{u}\| \leq \|\vec{w}\|$ for any $\vec{w} \in \mathcal{L}(B) \backslash \{\vec{0}\}$.

The corresponding problem is known as the Shortest Vector problem (SVP) and is one of the most famous problems related to lattices. We will use the notation SVP_p to denote the shortest vector problem with respect to l_p norm.

Remark 1.4.2. It is important to note here that the solution to the SVP depends on the underlying norm. Consider for example the lattice generated by the vectors

$$b_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \text{ and } b_2 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}.$$

Then clearly $[0,2]^T$ is a shortest vector (not the single one however) with respect to l_1 but not with respect to l_2 or l_∞. For the latter norms a shortest vector is $[1,1]^T$ which is shorter than $[0,2]^T$.

We now generalize the above definition.

Definition 1.4.3 (**Successive Minima** $\lambda_1, \lambda_2, ..., \lambda_n$)
Let $\|\cdot\|$ be an arbitrary norm. For every lattice $\mathcal{L} \subseteq \mathbb{R}^m$ of rank n the successive minima $\lambda_1, \lambda_2, ..., \lambda_n$ with respect to the norm $\|\cdot\|$ are defined as

$$\lambda_i(\mathcal{L}) := inf \left\{ r > 0 \;\middle|\; \begin{array}{l} \text{There are i lineary independent} \\ \text{vectors } c_1, c_2, ..., c_i \in \mathcal{L} \\ \text{with } \|c_j\| \leq r \text{ for } j = 1,2,...,i \end{array} \right\} \text{ for } i = 1,2,...,n \tag{1.12}$$

The above definition (visualized in fig 1.5) is due to Minkowski. From the

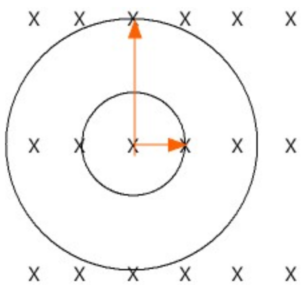

Figure 1.5: Successive Minima in a 2-dimensional lattice

definition given it is obvious that:

$$\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n.$$

In addition it is not difficult to derive the following upper bound for the successive minima:
For any lattice basis $b_1, b_2, ..., b_n$ and for $i = 1,2,...,n$

$$\max_{j=1,2,...,n} \|b_j\| \geq \lambda_i.$$

If the above inequality didn't hold then the linearly independent vectors $b_1, b_2, ..., b_n$ of the basis would form (not necessarily in the same order) a better set of successive minima which yields a contradiction. The following theorem gives a very useful lower bound on the length of the shortest nonzero vector in a lattice. We consider the Euclidean Norm.

Theorem 1.4.4
Let B be a basis of a lattice of rank n and B^* its Gram-Schmidt orthogonalization. Then

$$\lambda_1(\mathcal{L}(B)) \geq \min_{i=1,2,...,n} \|b_i^*\| > 0.$$

Proof. Note first that b_i^* are not (necessarily) lattice vectors.It suffices to show that the above inequality holds for every lattice vector. Since B forms a basis, every nonzero lattice vector $\vec{y}$ can be written as $\vec{y} = B\vec{x}$ where $\vec{x} \neq \vec{0}$. Let now $j \in \{1, 2, ..., n\}$ be the maximum index such that $x_j \neq 0$ (the requirement $\vec{x} \neq \vec{0}$ guarantees the existence of such a maximum). Then

$$|\langle Bx, b_j^* \rangle| = |\langle \sum_{i=1}^{j} x_i b_i, b_j^* \rangle| = |x_j| \langle b_j^*, b_j^* \rangle = |x_j| \|b_j^*\|^2$$

The second equality is obtained by the expression 1.4 from which it is obvious that $\langle b_i, b_j^* \rangle = 0$ for all $i < j$ and $\langle b_j, b_j^* \rangle = \langle b_j^*, b_j^* \rangle$.
In addition, Cauchy-Schwarz inequality implies that $|\langle Bx, b_j^* \rangle| \leq \|Bx\| \cdot \|b_j^*\|$ which finally yields

$$\|Bx\| \geq \frac{|\langle Bx, b_j^* \rangle|}{\|b_j^*\|} = |x_j| \|b_j^*\| \geq \|b_j^*\| \geq min\|b_i^*\|.$$

Since every vector of the lattice is at least "as long" as $min\|b_i^*\|$, obviously the above inequality also holds for the vector u which achieves the norm λ_1 and which, by definition, is a lattice vector. □

We now give a formal proof that the norm λ_1 is always achieved by some lattice vector.

Lemma 1.4.5. Let $\beta = \min_i \|b_i^*\|$ and $\vec{u}, \vec{w} \in \mathcal{L}$ be two lattice vectors. Then $\|\vec{u} - \vec{w}\| < \beta$ implies that $\vec{u} = \vec{w}$.

Proof. Let us assume for contradiction that $\vec{u} \neq \vec{w}$.Then $\vec{u} - \vec{w}$ is a nonzero lattice vector which, by theorem 1.4.4, means that $\|\vec{u} - \vec{w}\| \geq \beta$. Obviously this leads to a contradiction. □

Theorem 1.4.6
The Shortest Vector Problem (SVP) is well defined, in that there always exists a vector of minimal length which belongs to the lattice.

Proof. In order to prove the above theorem, we resort to the definition of the first minimum λ_1. By definition

$$\lambda_1 = \inf\{\|\vec{u}\| : \vec{u} \in \mathcal{L}(B) \backslash \{\vec{0}\}\}.$$

So there exists a sequence $\vec{u_i} \in \mathcal{L}$ such that

$$\lim_{i\to\infty} \|\vec{u_i}\| = \lambda_1.$$

Let assume wlog that u_i lies within a ball with center at $\vec{0}$ and radius $2\lambda_1$:

$$\vec{u_i} \in \mathcal{B}(0, 2\lambda_1) = \{\vec{y} : \|\vec{y}\| \leq 2\lambda_1\}.$$

The compactness of $\mathcal{B}(0, 2\lambda_1)$ implies the existence of a convergent subsequence $\{\vec{w_i}\} \subseteq \{\vec{u_i}\}$ such that $\lim_{i\to\infty} \|\vec{w_i}\| = \|\vec{w}\|$ for some vector $\vec{w}$. In addition

$$\|\vec{w}\| = \lim_{i\to\infty} \|\vec{w_i}\| = \lim_{i\to\infty} \|\vec{u_i}\| = \lambda_1.$$

Recall that $\vec{w_i}$ belong to the lattice (just as $\vec{u_i}$ do) and it only remains to prove that $\vec{w}$ is a lattice vector too. For that we need to observe that

$$\lim_{i\to\infty} \|\vec{w_i} - \vec{w}\| = 0$$

which by the definition of lim means that for "sufficiently" large i we have that $\|\vec{w_i} - \vec{w}\| < \beta/2$. The triangle inequality now gives (for "sufficiently" large i, j)

$$\|\vec{w_i} - \vec{w_j}\| \leq \|\vec{w_i} - \vec{w}\| + \|\vec{w} - \vec{w_j}\| < 2\frac{\beta}{2} = \beta.$$

We now invoke the previous lemma to conclude that $\vec{w_i} = \vec{w_j}$. This proves that for "sufficiently" large i the vectors $\vec{w_i}$ are identical and equal to their limit $\vec{w}$ which therefore belongs to the lattice. □

The successive minima λ_i depend on the underlying norm too.We can derive inequalities for the successive minima and for each of the known norms directly from the inequalities I1.For example, the norms l_2, l_∞ are related with the following inequality:

$$\lambda_{1,\infty}(L) \leq \lambda_{1,2}(L) \leq \sqrt{n} \cdot \lambda_{1,\infty}(L).$$

1.4.2 Minkowski's Convex Body Theorem

We will now concentrate our efforts on finding upper bounds on the length of the shortest vector in a lattice. We start by stating and prooving a very important theorem (Minkowski's Convex Body Theorem).In the next chapter we present efficient (polynomial) algorithms to obtain vectors that approximate the shortest vectors in a lattice.
We first prove the following lemma (due to Blichfeldt).

Lemma 1.4.7 (Blichfeldt Lemma). Let $S \subseteq \mathbb{R}^m$ be a set and $\mathcal{L}(B)$ a full dimensional lattice $(m = n)$ with base B. If

- $vol(S) > det(B)$ or
- $vol(S) = det(B)$ and S is compact

then there exist $\vec{z_1}, \vec{z_2} \in S$ (with $\vec{z_1} \neq \vec{z_2}$) such that $\vec{z_1} - \vec{z_2} \in \mathcal{L}(B)$.

Proof. We give the proof for the first case where $vol(S) > det(B)$.Using a compactness argument, we can similarly prove the case where S is compact and $vol(S) = det(B)$.

If B is the basis of $\mathcal{L}$ then as $\vec{x}$ ranges over the whole lattice, the sets $\vec{x}+\mathcal{P}(B) := \{\vec{x}+\vec{w} | \vec{w} \in \mathcal{P}(B)\}$ form a partition of $\mathbb{R}^n$. We now define the following sets

$$S_{\vec{x}} = S \cap (\vec{x} + \mathcal{P}(B)) \text{ where } \vec{x} \in \mathcal{L}(B).$$

Since the sets $\vec{x} + \mathcal{P}(B)$ form a $\mathbb{R}^n$ partition and $(\vec{x_1} + \mathcal{P}(B)) \cap (\vec{x_2} + \mathcal{P}(B)) = \emptyset$ for $\vec{x_1} \neq \vec{x_2}$ it is clear that the above sets form a partition of S, that is they are pairwise disjoint and

$$S = \bigcup_{\vec{x} \in \mathcal{L}(B)} S_{\vec{x}}.$$

In addition

$$vol(S) = \sum_{\vec{x} \in \mathcal{L}(B)} vol(S_{\vec{x}}).$$

We define the transtated sets $S_{\vec{x}} - \vec{x} = (S - \vec{x}) \cap \mathcal{P}(B)$. (by $S_{\vec{x}}$ we mean the set S' of all points z' where $z' = z - \vec{x}$ for all points $z \in S$). It is obvious that $vol(S_{\vec{x}}) = vol(S_{\vec{x}} - \vec{x})$. Thus

$$det(B) = vol(\mathcal{P}(B)) < vol(S) = \sum_{\vec{x} \in \mathcal{L}(B)} vol(S_{\vec{x}}) = \sum_{\vec{x} \in \mathcal{L}(B)} vol(S_{\vec{x}} - \vec{x}).$$

The facts that $S_{\vec{x}} - \vec{x} \subseteq \mathcal{P}(B)$ and $vol(\mathcal{P}(B)) < \sum_{\vec{x} \in \mathcal{L}(B)} vol(S_{\vec{x}} - \vec{x})$ imply that the sets $S_{\vec{x}} - \vec{x}$ are not mutually disjoint for all lattice vectors $\vec{x}$. That means that there exist $\vec{x}, \vec{y} \in \mathcal{L}(B)$ with $\vec{x} \neq \vec{y}$ such that $(S_{\vec{x}} - \vec{x}) \cap (S_{\vec{y}} - \vec{y}) \neq \emptyset$. Let $\vec{z} \in (S_{\vec{x}} - \vec{x}) \cap (S_{\vec{y}} - \vec{y})$. Then by definition of $S_{\vec{x}} - \vec{x}$ and $S_{\vec{y}} - \vec{y}$ the vectors

$$\begin{aligned} \vec{z_1} &= \vec{z} + \vec{x} \\ \vec{z_2} &= \vec{z} + \vec{y} \end{aligned}$$

belong to $S_{\vec{x}}$ and $S_{\vec{y}}$ respectively which in turn are subsets of S. Then

$$\vec{z_1} - \vec{z_2} = \vec{x} - \vec{y} \in \mathcal{L}(B)$$

and the proof is complete. □

We are now ready to prove Minkowski's Convex body theorem after we give the following definitions.

Definition 1.4.8 (**Symmetric Set**)
We say that a set S is *symmetric* (or more precisely *centrally symmetric* or *null-symmetric*) if for every $x \in S$, $-x \in S$ is true as well.

Definition 1.4.9 (**Convex Set**)
A set S is said to be *convex* if for any $x, y \in S$ and any $t \in [0, 1]$ we have that $tx + (1-t)y \in S$.

Theorem 1.4.10 (**Minkowski's Convex Body Theorem**)
Let $S \subseteq \mathbb{R}^n$ be a convex symmetric set and $\mathcal{L}(B)$ a full dimensional lattice ($m = n$) with base B. If

- $vol(S) > 2^n det(B)$ *or*
- $vol(S) = 2^n det(B)$ *and S is compact*

then S contains a nonzero lattice point.

Proof. We again give the proof for the first case. For the other case we just have to incorporate a compactness argument.
Let $S' = \frac{1}{2}S = \{x | 2x \in S\}$. Then obviously $vol(S') = 2^{-n}vol(S) > det(\mathcal{L})$ by hypothesis. By lemma 1.4.7 there exist two (distinct) points $z_1, z_2 \in S'$ such that $z_1 - z_2 \in \mathcal{L}$ is a nonzero lattice point. We will now prove that $z_1 - z_2$ belongs to S. Notice that by definition of S' both $2z_1, 2z_2$ belong to S and so does $-2z_2$ because S is symmetric. Finally the fact that S is convex, implies that $\frac{1}{2}(2z_1) + \frac{1}{2}(-2z_2) = z_1 - z_2$ belongs to S and this completes the proof. □

Remark 1.4.11. The fact that S is symmetric implies that $z_2 - z_1$ is also nonzero and belongs to S. We can then restate Minkowski's Convex Body Theorem as follows:

Theorem 1.4.12
Let $\mathcal{L} \subseteq \mathbb{R}^n$ be a full dimensional lattice and $S \subseteq \mathbb{R}^n$ a convex, symmetric, compact set with $vol(S) \geq 2^n det(\mathcal{L})$. Then $|S \cap \mathcal{L}| \geq 3$, that is, S contains at least two nonzero vectors $\pm\vec{u} \in \mathcal{L}$.

The following corollary demonstartes the relation between the above theorem and bounding the length of the shortest vector in a lattice.

Corollary 1.4.13. For all full dimensional lattices $\mathcal{L}(B)$, there exists a lattice point $x \in \mathcal{L}(B) \backslash \{0\}$ such that

$$\|x\|_\infty \leq \sqrt[n]{det(\mathcal{L})}. \tag{1.13}$$

Proof. Consider the set S defined as

$$S = \{\vec{x} : \|\vec{x}\|_\infty \leq \sqrt[n]{det(\mathcal{L})}\}.$$

Apparently, S is symmetric and convex and in addition $vol(S) = 2^n det(\mathcal{L})$. Minkowski's Theorem then guarantees that there exists a vector $\vec{v} \in \mathcal{L}(B)$ such that

$$\|\vec{v}\|_\infty \leq \sqrt[n]{det(\mathcal{L})}.$$

□

Below we summarize the previous results giving two very important inequalities for the norms l_2, l_∞ of the shortest vector of a lattice. The second inequality is an immediate result of the first and the inequalities I1.We claim that for all full dimensional lattices ($n = m$) there exist lattice points (not necessarily identical) $x, y \neq 0$ such that

$$\|x\|_\infty \leq \sqrt[n]{det(\mathcal{L})} \text{ and} \tag{SVI1}$$

$$\|y\|_2 \leq \sqrt{n}\,\sqrt[n]{det(\mathcal{L})}. \tag{SVI2}$$

Actually the last inequality is strict. In order to prove this, we first need the following lemma.

Lemma 1.4.14. The volume of an n-dimensional ball of radius r is

$$vol(\mathcal{B}(0,r)) > (\frac{2r}{\sqrt{n}})^n.$$

Proof. It is easy to see that each ball of radius r contains a cube of side length $\frac{2r}{\sqrt{n}}$. Thus

$$\{x \in \mathbb{R}^n | \forall i, |x_i| < \frac{r}{\sqrt{n}}\} \subset \mathcal{B}(0,r)$$

which means that $vol(\mathcal{B}(0,r)) > vol(cube) = (\frac{2r}{\sqrt{n}})^n$. □

Theorem 1.4.15
For any full rank lattice $\mathcal{L}$ of rank n

$$\lambda_1(\mathcal{L}) < \sqrt{n}(det(\mathcal{L}))^{\frac{1}{n}}.$$

Proof. Consider the (open) ball $\mathcal{B}(0, \lambda_1(\mathcal{L}))$ which by definition contains no nonzero lattice points. Then theorem 1.4.10 and lemma 1.4.14 imply that

$$(\frac{2\lambda_1(\mathcal{L})}{\sqrt{n}})^n < vol(\mathcal{B}(0, \lambda_1(\mathcal{L}))) \leq 2^n \cdot det(\mathcal{L}).$$

Thus $\lambda_1(\mathcal{L}) < \sqrt{n}(det(\mathcal{L}))^{\frac{1}{n}}$. □

The above inequality interestingly holds even if the lattice is not full rank ($n < m$). In order to prove that we only have to reduce the case $n < m$ to the case where $n = m$. In this thesis however, we will only use full rank lattices so the properties established are sufficient for the reader to follow the material presented in the next chapters.

Remark 1.4.16. Theorem 1.3.4 says that the determinant of a lattice is independent of the specific basis we use to produce it. Thus, in all the above theorems and definitions when we wrote $det(\mathcal{L})$ or $det(B)$ we meant the exact same thing. This should not cause any confusion to the reader.

Remark 1.4.17. It is important to note here that the results presented above for the shortest vectors in a lattice with respect to norms $\|\cdot\|_2$ and $\|\cdot\|_\infty$, only guarantee the existence of such short vectors and do not provide any efficient algorithm to actually construct them. In the next chapter we present an algorithm that can produce short vectors efficiently. These vectors, however, satisfy inequalities that are weaker than those already presented.

1.4.3 A Number Theoretic Application

We finish this chapter by presenting a number theoretic application of Minkowski's Convex Body Theorem. The following example, though trivial, illustrates how one can use lattices and their properties in order to prove theorems related to number theory.

Theorem 1.4.18
For every prime $p \equiv 1 \ (mod\ 4)$, there exist $a, b \in \mathbb{Z}$ such that $p = a^2 + b^2$.

Proof. By hypothesis, $p \equiv 1 \ (mod\ 4) \Rightarrow 4/p-1$. Thus $\frac{p-1}{2} = 2k\,, k \in \mathbb{Z}$ which means that $(-1)^{\frac{p-1}{2}} \equiv 1\,(mod p)$ and so -1 is a quadratic residue modulo p. Let i such that $i^2 \equiv -1\,(mod p)$. Then

$$p/i^2 + 1. \tag{1.14}$$

Consider now the lattice basis

$$\begin{bmatrix} 1 & 0 \\ i & p \end{bmatrix}.$$

Then Minkowski's Theorem (theorem 1.4.15) says that $SVP_2 < \sqrt{2} \cdot \sqrt{det(B)}$ or equivalently there exists an integer vector $\vec{x} = [x_1, x_2]^T$ such that

$$\|B\vec{x}\|_2 < \sqrt{2} \cdot \sqrt{det(B)} \Rightarrow \|B\vec{x}\|_2^2 < 2 \cdot det(B) = 2p.$$

If we expand the term $\|B\vec{x}\|_2^2$ we get

$$\|B\vec{x}\|_2^2 = x_1^2 + (ix_1 + px_2)^2.$$

Let $a = x_1$ and $b = ix_1 + px_2$. Clearly $(a, b) \neq (0, 0)$ (otherwise $x_1 = x_2 = 0$). This, combined with the previous inequality, gives that

$$0 < a^2 + b^2 < 2p. \tag{1.15}$$

We also have that

$$x_1^2 + (ix_1 + px_2)^2 = x_1^2 + i^2x_1^2 + p^2x_2^2 + 2ix_1px_2 = p(px_2^2 + 2ix_1x_2) + x_1^2(i^2 + 1)$$

where p divides the first term as well as the second because of condition 1.14. Thus $a^2 + b^2 = kp$ for some $k \in \mathbb{Z}$ which along with 1.15 finally gives

$$a^2 + b^2 = p.$$

□

Chapter 2

Lattice Basis Reduction

Consider the lattices produced by the following bases:

$$B_1 = \begin{bmatrix} 3 & 2 \\ 13 & 9 \end{bmatrix} \qquad and \qquad B_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The above bases are equivalent, that is they produce the exact same lattice (in particular $\mathbb{Z}^2$). Indeed one can produce B_1 by multiplying B_2 with the unimodular matrix

$$U = \begin{bmatrix} 3 & 2 \\ 13 & 9 \end{bmatrix} \qquad with \qquad U^{-1} = \begin{bmatrix} 9 & -2 \\ -13 & 3 \end{bmatrix}.$$

However it seems that B_2 is a more "elegant" description of the lattice. This is both because B_2 consists of smaller vectors and because it makes clear that $\mathbb{Z}^2$ is the lattice produced.

The above example leads to the observation that some bases are "better" than other bases of the same lattice. What we mean by "better" greatly depends on the actual application. In applications where lattices are used, we are mostly interested in bases made up of short vectors. We define the "better" basis by the more formal definition *reduced*. Consequently, *lattice basis reduction* is the process in which a reduced basis is found from a given basis. In this section we give various notions for reduction along with the algorithms that produce them.

All notions entail finding "sufficiently short" vectors of a lattice. Since we cannot find the shortest vector efficiently in the general case (n-dimensional lattice) we are searching for alternative algorithms that efficiently produce vectors that "adequately" approximate the shortest vector. The most popular such algorithm is LLL presented and extensively analyzed in section 2.3.

2.1 Minkowski Reduction

One of the first notions of reduction of a lattice basis was *Minkowski Reduction.*Recall that by definition 1.4.3 the i-th minimum of $\mathcal{L}$, $\lambda_i(\mathcal{L})$, is defined as the length of the shortest vector that is linearly independent from the $i-1$ vectors that give the previous minima. Unfortunately, a set of i vectors giving the first i minima cannot always be extended to form a basis of $\mathcal{L}$ for dimensions greater than 2. Instead, Minkowski Reduction requires a somewhat weaker condition.

Definition 2.1.1 (**Minkowski Reduction**)
Let $\mathcal{L}$ be a lattice with basis $B = [b_1, b_2, ..., b_n]$. We say that B is *Minkowski Reduced* if the following properties hold:

- b_1 is a shortest vector in $\mathcal{L}$ and, inductively,
- b_i is a shortest vector independent from $b_1, ..., b_{i-1}$ such that $b_1, b_2, ..., b_i$ can be extended to form a basis of $\mathcal{L}$.

Remark 2.1.2. The above reduction yields bases the vectors of which (and especially b_1 which is by definition a shorter vector in $\mathcal{L}$) enjoy important properties. Such kind of reduction has found application in the theory and geometry of numbers. However it has little computational worth since there is no known efficient (polynomial) algorithm that computes such a basis. While Minkowski Reduction can be used to prove some (existence) theorems in theory and geometry of numbers, it is not used in Cryptography where computational cost is a major concern. In the sections to come we present reduction algorithms that yield a weaker reduction but are computationally efficient.

2.2 Two-Dimensional (Gauss) Reduction

While there is no efficient algorithm that computes the shortest vectors in an n-dimensional lattice, for small fixed dimensions we can find polynomial-time algorithms that locate short vectors.

We study below the two-dimensional case and analyze a polynomial algorithm that given an arbitrary basis of a lattice, produces a reduced basis made up of the shortest vectors of the same lattice. This algorithm is a slight variation of an algorithm known as Gauss Reduction Algorithm and is quite similar to the Euclidean algorithm for the great common divisor. Before we present the algorithm and its analysis, we first give some definitions and theorems.

2.2.1 Definitions

Definition 2.2.1 (**Reduced Basis**)
Let $[\vec{b_1}, \vec{b_2}]$ be a basis of a lattice. We say that this basis is reduced if

$$\|\vec{b_1}\|, \|\vec{b_2}\| \leq \|\vec{b_1} + \vec{b_2}\|, \|\vec{b_1} - \vec{b_2}\|. \tag{2.1}$$

A geometrical interpretation of the above definition is that both sides of the fundamental parallelipiped associated to the basis are shorter than its diagonals.

Remark 2.2.2. Swapping (if needed) vectors $\vec{b_1}, \vec{b_2}$ to satisfy $\|\vec{b_1}\| \leq \|\vec{b_2}\|$ or changing the sign of vector $\vec{b_2}$ in order to achieve $\|\vec{b_1} - \vec{b_2}\| \leq \|\vec{b_1} + \vec{b_2}\|$, the condition of the above definition can be rewritten in the following form:

$$\|\vec{b_1}\| \leq \|\vec{b_2}\| \leq \|\vec{b_1} - \vec{b_2}\| \leq \|\vec{b_1} + \vec{b_2}\|.$$

Before we proceed, we need the following lemma.

Lemma 2.2.3. Let $\vec{x}, \vec{x} + \vec{y}, \vec{x} + r\vec{y}$ be three vectors in a line and $r \in [1, +\infty)$. Then if $\|\vec{x}\| \leq \|\vec{x} + \vec{y}\|$ then $\|\vec{x} + \vec{y}\| \leq \|\vec{x} + r\vec{y}\|$.

Proof. Define $s = \frac{1}{r} \in (0, 1]$. Then

$$\vec{x} + \vec{y} = (1 - s)\vec{x} + s(\vec{x} + r\vec{y})$$

and by the triangle inequality we have

$$\|\vec{x} + \vec{y}\| \leq (1 - s)\|\vec{x}\| + s\|\vec{x} + r\vec{y}\| \leq (1 - s)\|\vec{x} + \vec{y}\| + s\|\vec{x} + r\vec{y}\|.$$

The second inequality is dictated by the hypothesis $\|\vec{x}\| \leq \|\vec{x} + \vec{y}\|$. We conclude that

$$\|\vec{x}+\vec{y}\| \leq (1-s)\|\vec{x}+\vec{y}\|+s\|\vec{x}+r\vec{y}\| \Rightarrow s\|\vec{x}+\vec{y}\| \leq s\|\vec{x}+r\vec{y}\| \Rightarrow \|\vec{x}+\vec{y}\| \leq \|\vec{x}+r\vec{y}\|.$$

□

We now establish the equivalence between the basis vectors of a two-dimensional reduced basis and the successive minima of a lattice.

Theorem 2.2.4
Let $[\vec{b_1}, \vec{b_2}]$ be the basis of a lattice. Then $\|\vec{b_1}\| = \lambda_1$ and $\|\vec{b_2}\| = \lambda_2$ if and only if $\|\vec{b_1}\| \leq \|\vec{b_2}\| \leq \|\vec{b_1} + \vec{b_2}\|, \|\vec{b_1} - \vec{b_2}\|$.

Proof. ($\Rightarrow$) Let $\|\vec{b_1}\| = \lambda_1$ and $\|\vec{b_2}\| = \lambda_2$. Then by definition $\vec{b_1}$ is the shortest vector in the lattice. So obviously $\|\vec{b_1}\| \leq \|\vec{b_2}\|, \|\vec{b_1} + \vec{b_2}\|, \|\vec{b_1} - \vec{b_2}\|$. In addition suppose that $\|\vec{b_1} + \vec{b_2}\| < \|\vec{b_2}\|$. Then the linear independence of $\vec{b_1}, \vec{b_1} + \vec{b_2}$ would imply that $\lambda_2 \leq \|\vec{b_1} + \vec{b_2}\| < \|\vec{b_2}\|$ which yields a contradiction. Thus $\|\vec{b_1} + \vec{b_2}\| \geq \|\vec{b_2}\|$. A similar argument works for $\vec{b_1} - \vec{b_2}$ and this completes this direction of the proof.
($\Leftarrow$) Let now $\|\vec{b_1}\| \leq \|\vec{b_2}\| \leq \|\vec{b_1} + \vec{b_2}\|, \|\vec{b_1} - \vec{b_2}\|$. In order to show that $\|\vec{b_1}\| = \lambda_1$ and $\|\vec{b_2}\| = \lambda_2$, it suffices to show that for every $r, s \in \mathbb{Z}$

1. $\|\vec{b_1}\| \leq \|r\vec{b_1} + s\vec{b_2}\|$ for $(r, s) \neq (0, 0)$ and
2. $\|\vec{b_2}\| \leq \|r\vec{b_1} + s\vec{b_2}\|$ for $s \neq 0$ (the case $s = 0$ is excluded since $\vec{b_1}, \vec{b_2}$ are linearly independent).

We consider 3 cases:

(a) $r = 0$: Then $s \neq 0$ and $\|\vec{b_1}\| \leq \|\vec{b_2}\| \leq \|s\vec{b_2}\| = \|r\vec{b_1} + s\vec{b_2}\|$.

(b) $s = 0$: Then $r \neq 0$ and $\|\vec{b_1}\| \leq \|r\vec{b_1}\| = \|r\vec{b_1} + s\vec{b_2}\|$.

(c) $r, s \neq 0$: Assume wlog that $r \geq s \geq 0$.Then

$$\|\vec{b_2} + \frac{r}{s}\vec{b_1}\| = \|\frac{r\vec{b_1} + s\vec{b_2}}{s}\| \leq \|r\vec{b_1} + s\vec{b_2}\|.$$

We now apply lemma 2.2.3 to the vectors $\vec{b_2}, \vec{b_2}+\vec{b_1}\, and\, \vec{b_2}+\frac{r}{s}\vec{b_1}$ where $\frac{r}{s} \geq 1$ to get

$$\|\vec{b_2}\| \leq \|\vec{b_1} + \vec{b_2}\| \leq \|\vec{b_2} + \frac{r}{s}\vec{b_1}\| \leq \|r\vec{b_1} + s\vec{b_2}\|.$$

All the other cases for r, s can be proved in a similar fashion.

This completes our proof that the vectors of a reduced basis of a two-dimensional lattice are the shortest possible. □

We give below (algorithm 2) the initial algorithm for the l_2 norm known as *Gauss Reduction Algorithm* and then present an analyze the extension of Gauss Algorithm due to Kaib and Schnorr that works for arbitrary norms. The generalized lattice algorithm presented in algorithm 3 works with well ordered bases until a reduced basis is found. Then theorem 2.2.4 guarantees that we have found the shortest vectors in the lattice.

Algorithm 2: Gaussian Reduction (b_1, b_2)

```
Input: A basis $B = [b_1, b_2]$.
Output: A Gaussian reduced Basis $B' = [b'_1, b'_2]$.
begin
    repeat
        if ($\|b_1\| > \|b_2\|$) then
            swap $b_1, b_2$;
        end
        $\mu \leftarrow \langle b_1, b_2\rangle / \|b_1\|^2$;
        $b_2 \leftarrow b_2 - \lceil \mu \rfloor b_1$;
        (where $\lceil a \rfloor = \lfloor a + 0.5 \rfloor$)
    until ($\|b_1\| < \|b_2\|$) ;
    output $(b_1, b_2)$
end
```

Definition 2.2.5 (**Well Ordered Basis**)
A basis $[\vec{b_1}, \vec{b_2}]$ is said to be *well ordered* if

$$\|\vec{b_1}\| \leq \|\vec{b_1} - \vec{b_2}\| < \|\vec{b_2}\|. \tag{2.2}$$

Remark 2.2.6. The above definition immediately implies that

$$\|\vec{b_1}\| \leq \|\vec{b_1} - \vec{b_2}\| < \|\vec{b_2}\| \leq \|\vec{b_1} + \vec{b_2}\|. \tag{2.3}$$

We can see that if we apply lemma 2.2.3 to vectors $\vec{b_2} - \vec{b_1}, \vec{b_2}$ and $\vec{b_1} + \vec{b_2}$ which are on the same line.

Below (algorithm 3) we provide the generalized Gauss Reduction for arbitrary norms. In the above algorithm we can always assume that the input basis $[\vec{b_1}, \vec{b_2}]$ is

Algorithm 3: Generalized Gaussian Reduction (b_1, b_2) for Arbitrary Norm

```
Input: A well-ordered basis $B = [b_1, b_2]$.
Output: A Reduced Basis $B' = [b'_1, b'_2]$.
(The algorithm works for any norm)
begin
    while ($\|b_2\| > \|b_1 - b_2\|$) do
        1.1 $b_2 \leftarrow b_2 - \mu b_1$,   $\mu \in \mathbb{Z}$ chosen so that $\|b_2 - \mu b_1\|$ is
        minimized ;
        1.2 if ($\|b_1 + b_2\| < \|b_1 - b_2\|$) then
            $b_2 \leftarrow -b_2$;
        end
        1.3 exchange $b_1, b_2$ ;
    end
    output Reduced Basis $(b_1, b_2)$
end
```

well ordered (if not reduced). That is a reasonable assumption since we can always produce a well ordered basis by swapping $\vec{b_1}, \vec{b_2}$ or replacing $\vec{b_2}$ by $\vec{b_1} - \vec{b_2}$ or $\vec{b_1} + \vec{b_2}$ accordingly.

Remark 2.2.7. The generalized Gauss Reduction Algorithm expects a well-ordered basis. We claim that each time the algorithm enters the while loop, the current basis is well-ordered.
This is indeed the case at the beginning of the algorithm as we mentioned earlier. If the algorithm enters the loop again it means that it has failed to produce a reduced basis at the last step (Step 1.3) of the previous while loop. The basis computed at that step is

$$\vec{b'_1} = \pm(\vec{b_2} - \mu\vec{b_1}) \text{ and } \vec{b'_2} = \vec{b_1}$$

which means that

$$\|\vec{b'_1} - \vec{b'_2}\| = \| \pm (\vec{b_2} - \mu\vec{b_1}) - \vec{b_1}\| = \|\vec{b_2} - (\mu \pm 1)\vec{b_1}\| \geq \|\vec{b_2} - \mu\vec{b_1}\| = \|\vec{b'_1}\|.$$

The last inequality is true because the value μ computed by the algorithm in step 1.1 renders the minimal $\|\vec{b_2} - \mu\vec{b_1}\|$ over all integer values of μ. Similarly $\|\vec{b_1'} + \vec{b_2'}\| \geq \|\vec{b_1}\|$. Finally $\|\vec{b_1'} - \vec{b_2'}\| \leq \|\vec{b_2}\|$ otherwise the basis would be reduced. We have therefore proved that the basis given as input in the beginning of each while loop is well ordered.

In the following lines we will prove the correctness of the above algorithm and analyze its performance.

2.2.2 Correctness

In order to prove the correctness we have to show both that the algorithm performs the required task when it terminates (that is, it produces a reduced basis) and that it actually terminates.

We proved in the previous remark that each time the algorithm enters the while loop the basis is well ordered. So the termination of the algorithm would mean that the while condition $\|\vec{b_1} - \vec{b_2}\| < \|\vec{b_2}\|$ is not true.This means that $\|\vec{b_2}\| \leq \|\vec{b_1} - \vec{b_2}\| \leq \|\vec{b_1} + \vec{b_2}\|$. The last inequality holds because of step 1.2 of the algorithm. This means that the basis $[\vec{b_1}, \vec{b_2}]$ is reduced upon termination. In addition, the algorithm only performs elementary column operations which means that the basis produced by the algorithm is indeed equivalent to the initial basis.

As far as termination is concerned, recall that at the end of step 1.3 (and while the basis is not yet reduced) we have $\|\vec{b_1'}\| \leq \|\vec{b_2'}\| = \|\vec{b_1}\|$. As the value of $\|\vec{b_1}\|$ gets smaller at every iteration of the algorithm, we can see (using a compactness argument) that it will at some point reach its minimum. At this point the algorithm terminates.

2.2.3 Running Time Analysis

The following analysis shows that the algorithm is polynomial in its input size. This means that its running time is logarithmic in the lenght of the initial basis vectors.

Clearly the operations performed by the algorithm in each iteration are polynomial in the input size. Thus, it only remains to show that the number of iterations is polynomial too. Let k be the total number of iterations. We will prove that k is polynomially bounded by the size of the input of the algorithm, or, otherwise stated, that k is bounded by $p(\log\|\vec{b_1}\|)$ where p is a polynomial.

For that suppose that in the beggining of the algorithm $\vec{b_1} = \vec{u_k}$ and $\vec{b_2} = u_{\vec{k+1}}$. At the end of the k-th iteration we get the reduced basis $\vec{u_1}, \vec{u_2}$. We will focus on the successive values of $\vec{b_1}$ in each step. $\vec{u_i}$ will denote $\vec{b_1}$ in the $(k-i+1)$-th iteration. As the algorithm runs we get the sequence $(\vec{u_k}, ..., \vec{u_2}, \vec{u_1})$ (in chronological order). We need the following lemma.

Lemma 2.2.8. $\forall i > 1\,,\ \|\vec{u_i}\| \leq \frac{1}{2}\|u_{\vec{i+1}}\|$.

Proof. Consider the sequence $(\vec{u_{i-1}}, \vec{u_i}, \vec{u_{i+1}})$. For simplicity's sake we will use instead the vectors $(\vec{x}, \vec{y}, \vec{z})$.Both $[\vec{x}, \vec{y}]$ and $[\vec{y}, \vec{z}]$ are well ordered.Let $s = \pm 1$. We know by the analysis above that $\vec{x} = s(\vec{z} - \mu\vec{y})$ which gives (multiply both sides with s) $\vec{z} = s\vec{x} + \mu\vec{y}$. We consider the following cases.

(a) ($\mu = 0$ or $\mu = 1$).Not possible. Both would imply that $[\vec{y}, \vec{z}]$ is not well ordered. The first condition would give $\|\vec{z}\| = \|\vec{x}\| < \|\vec{y}\|$, while the second would lead to $\|\vec{z} - \vec{y}\| = \|\vec{x}\| < \|\vec{y}\|$.

(b) ($s = -1, \mu = 2$). Then $\|\vec{z} - \vec{y}\| = \| - \vec{x} + \vec{y}\| < \|\vec{y}\|$, which also contradicts the fact that $[\vec{y}, \vec{z}]$ is well ordered.

(c) ($s = -1, \mu > 2$).We know that $\|\vec{z}\| = \|-\vec{x}+\mu\vec{y}\| \geq \mu\|\vec{y}\|-\|\vec{x}\| \geq \mu\|\vec{y}\|-\|\vec{y}\| = (\mu - 1)\|\vec{y}\| \geq 2\|\vec{y}\|, \Rightarrow \|\vec{y}\| \leq \frac{1}{2}\|\vec{z}\|$.

(d) ($s = 1, \mu \geq 2$). $\|2\vec{y}-\vec{x}\| \leq \|\vec{y}\|+\|\vec{y}-\vec{x}\| \leq \|\vec{y}\|+\|\vec{y}\| = 2\|\vec{y}\|$. We now apply lemma 2.2.3 to the vectors $2\vec{y} - \vec{x}, 2\vec{y}, 2\vec{y} + \vec{x}$ and we get

$$\|2\vec{y}\| \leq \|2\vec{y} + \vec{x}\|.$$

Since $[\vec{x}, \vec{y}]$ is well ordered, we know that $\|\vec{x}\| < \|\vec{y} + \vec{x}\|$ and consequently (again by lemma 2.2.3)

$$\|\vec{y} + \vec{x}\| < \|2\vec{y} + \vec{x}\| \leq \|\mu\vec{y} + \vec{x}\|.$$

We then have

$$\|\vec{z}\| = \|\mu\vec{y} + \vec{x}\| \geq \|2\vec{y} + \vec{x}\| \geq 2\|\vec{y}\|.$$

This completes the proof as we have covered all possible cases. □

We are now ready to establish the main theorem for the running time of the Generalized Gauss Algorithm.

Theorem 2.2.9

$$\forall i > 1\,,\ \|\vec{u_i}\| \geq 2^{i-1}\|\vec{u_1}\|. \tag{2.4}$$

Proof. Trivial.We use induction and the previous lemma. □

Corollary 2.2.10. The number of iterations of the Generalized Gauss Algorithm is bounded by $\log_2 \|\vec{b_1}\|$.

Proof. By the previous theorem we have $\|\vec{b_1}\| = \|\vec{u_k}\| \geq 2^{k-1}\|\vec{u_1}\|$. If we take the logarithm (with respect to 2) of both sides we get the desired bound. □

Remark 2.2.11. We just mention here that the above algorithm has been generalized by Nguyen and Stelhe to lattices of any dimension. However, this greedy algorithm is optimal (that is, it produces vectors b_i such that $b_i = \lambda_i(\mathcal{L})$ for each $i = 1, 2, ..., m$) only for lattices of dimension $m \leq 4$.

2.3 LLL Reduction

In section 2.2 we presented an algorithm that finds in polynomial time a reduced basis for a two-dimensional lattice. This reduced basis consisted of vectors $[\vec{b_1}, \vec{b_2}]$ such that $\|\vec{b_1}\| = \lambda_1$ and $\|\vec{b_2}\| = \lambda_2$. We also mentioned that there are similar algorithms that efficiently find a reduced basis in lattices with dimension 3 and 4. We want to generalize the previous procedures for arbitrary dimensions. However, up to now, there is no known algorithm that finds the shortest vector in a n-dimensional lattice. In this section we will concentrate our attention to a very famous approximation algorithm, LLL algorithm, named after its inventors (Lenstra, Lenstra, Lovasz) which does not necessarily find the shortest vector in a lattice, but computes a lattice vector that is provably at most $c(n)$ times the length of the shortest vector, where $c(n)$ is a function of the lattice dimension n.

We divide this section in five subsections. In subsection 2.3.1 we introduce a new notion for basis reduction (LLL Reduction) and prove some properties that the vectors of such a basis satisfy. In subsection 2.3.2 we provide an algorithm that actually computes an LLL Reduced Basis and prove its correctness. In subsection 2.3.3 we analyze the running time of the new algorithm and prove that it runs in polynomial time. Next (subsection 2.3.4) we present how one can use LLL in order to find a solution to the Simultaneous Diophantine Approximation Problem. Finally in subsection 2.3.5 we quote some other applications of the new algorithm.

2.3.1 Definitions and Properties of LLL Reduction

We first define projection operations π_i from $\mathbb{R}^m$ onto $\sum_{j\geq i} \mathbb{R}\vec{b_j^*}$:

$$\pi_i(\vec{x}) = \sum_{j=1}^{n} \frac{\langle \vec{x}, \vec{b_j^*}\rangle}{\langle \vec{b_j^*}, \vec{b_j^*}\rangle} \vec{b_j^*}. \tag{2.5}$$

The operator π_i will help us define a new notion for basis reduction.The definition of π_i operator implies that when we apply π_i to a vector $\vec{x} \in \mathbb{R}^m$ we get only the components of $\vec{x}$ that are perpendicular to the space spanned by $\vec{b_1^*}, ..., \vec{b_{i-1}^*}$ or equivalently, only the components of $\vec{x}$ that live in the space spanned by $\vec{b_i^*}, ..., \vec{b_n^*}$. We can now define LLL Reduction as follows:

Definition 2.3.1 (LLL Reduced Basis)
A basis $B = [\vec{b_1}, \vec{b_2}, ..., \vec{b_n}] \in \mathbb{R}^{m\times n}$ is said to be **LLL Reduced** with parameter δ ($\frac{1}{4} < \delta \leq 1$) if:

1. $|\mu_{i,j}| \leq \frac{1}{2} \quad \forall i > j$ where $\mu_{i,j}$ denote the Gram-Schmidt coefficients.

2. for any pair of consecutive vectors $\vec{b_i}, \vec{b_{i+1}}$ we have that

$$\delta \|\pi_i(\vec{b_i})\|^2 \leq \|\pi_i(\vec{b_{i+1}})\|^2 \tag{2.6}$$

or expressed in another way

$$\delta\|\vec{b_i^*}\|^2 \leq \|\vec{b_{i+1}^*} + \mu_{i+1,i} \cdot \vec{b_i^*}\|^2. \tag{2.7}$$

Remark 2.3.2. To see that the two expressions of the second condition are equivalent, one should just recall the Gram-Scmidt Orthogonalization Procedure (and more specifically equation 1.4).This, along with the fact that $\langle \vec{b_i^*}, \vec{b_j^*} \rangle = 0 \quad \forall i \neq j$ and that operator π_i contains only the components of $\vec{b_i}, \vec{b_{i+1}}$ that are perpendicular to $\vec{b_1^*}, ..., \vec{b_{i-1}^*}$, yield the above equivalence.

Remark 2.3.3. Let us now take a more thorough look at the two conditions. The first one guarantees that the basis produced is *Length Reduced* which means that the final vectors are "as close as possible" to the vectors Gram-Schmidt Orthogonalization produces. This will be further clarified when we present the LLL algorithm. The second condition guarantees that at the end of the algorithm , the vector $\vec{b_1}$ of the reduced basis will be "small enough" to approximate the shortest vector of the lattice.

Remark 2.3.4. In the above definition if we let $\delta = 1$ then the two conditions simply say that we require that the 2-dimensional basis $\pi_i([\vec{b_i}, \vec{b_{i+1}}])$ is reduced.

The above notion for basis reduction may seem a little weird at first glance. However, an LLL Reduced basis enjoys some very important properties and has found a vast number of applications in various fields. More sinificantly, as we shall see in the next subsection, there exists a polynomial time algorithm that, given an arbitrary basis of dimension n, can produce an equivalent LLL Reduced Basis.

The following theorem gives an approximation factor for the shortest vector in an LLL Reduced Basis with parameter δ.

Theorem 2.3.5
Let $\vec{b_1}, \vec{b_2}, ..., \vec{b_n} \in \mathbb{R}^n$ be an LLL Reduced Basis with parameter δ. Then

$$\|\vec{b_1}\| \leq (\frac{2}{\sqrt{4\delta - 1}})^{n-1}\lambda_1(\mathcal{L}). \tag{2.8}$$

Proof. For all $i = 1, 2, ...n$ we have

$$\begin{aligned}\delta\|\vec{b_i^*}\|^2 &\leq \|\vec{b_{i+1}^*} + \mu_{i+1,i} \cdot \vec{b_i^*}\|^2 \\ &= \|\vec{b_{i+1}^*}\|^2 + \|\mu_{i+1,i} \cdot \vec{b_i^*}\|^2 \\ &= \|\vec{b_{i+1}^*}\|^2 + |\mu_{i+1,i}|^2 \cdot \|\vec{b_i^*}\|^2 \\ &\leq \|\vec{b_{i+1}^*}\|^2 + \frac{1}{4}\|\vec{b_i^*}\|^2.\end{aligned}$$

This finally gives

$$(\delta - \frac{1}{4})\|\vec{b_i^*}\|^2 \leq \|\vec{b_{i+1}^*}\|^2. \tag{2.9}$$

In addition, by induction we have that

$$\|\vec{b_i^*}\|^2 \geq (\delta - \frac{1}{4})^{i-1}\|\vec{b_1^*}\|^2.$$

But $\vec{b_1^*} = \vec{b_1}$ and we have already shown that for any basis $\vec{b_1}, \vec{b_2}, ..., \vec{b_n}$, $\lambda_1(\mathcal{L}) \geq \min_i \|\vec{b_i^*}\|$. So for the i that yields the min $\|\vec{b_i^*}\|$

$$\begin{aligned}\|\vec{b_1}\| &\leq (\delta - \frac{1}{4})^{\frac{1-i}{2}}\|\vec{b_i^*}\| \leq (\delta - \frac{1}{4})^{\frac{1-n}{2}}\|\vec{b_i^*}\| \\ &\leq (\delta - \frac{1}{4})^{\frac{1-n}{2}}\lambda_1(\mathcal{L}) = (\frac{2}{\sqrt{4\delta - 1}})^{n-1}\lambda_1(\mathcal{L}).\end{aligned}$$

□

Remark 2.3.6. The paremeter δ can be any real number in the interval $(\frac{1}{4}, 1]$ as mentioned in the original LLL paper [24].However , the typical value used in almost every application is $\delta = \frac{3}{4}$. We will exclusively use this value from now on, in all the applications presented in following chapters.

Below we give a set of useful inequalities derived from the LLL reduction definition. We mention that the following inequalities (obtained for the specific value $\delta = \frac{3}{4}$) are possibly the most valuable result of the LLL Reduction for the needs of this thesis. These inequalities (and especially the last one) will be frequently invoked throughout the following chapters and therefore it is important for the reader to fully understand them.

Theorem 2.3.7
Let $\vec{b_1}, \vec{b_2}, ..., \vec{b_n}$ be an LLL reduced basis for a lattice $\mathcal{L} \in \mathbb{R}^n$ and $\vec{b_1}^*, \vec{b_2}^*, ..., \vec{b_n}^*$ the corresponding Gram-Schmidt vectors. Then we have:

$$\|\vec{b_j}\|^2 \leq 2^{i-1} \cdot \|\vec{b_i^*}\|^2 \qquad for\, 1 \leq j \leq i \leq n, \tag{LLL1}$$

$$\|\vec{b_1}\| \leq 2^{\frac{n-1}{2}}\lambda_1(\mathcal{L}), \tag{LLL2}$$

$$det(\mathcal{L}) \leq \prod_{i=1}^{n}\|\vec{b_i}\| \leq 2^{\frac{n(n-1)}{4}} \cdot det(\mathcal{L}), \tag{LLL3}$$

$$\|\vec{b_1}\| \leq 2^{\frac{n-1}{4}} \cdot det(\mathcal{L})^{\frac{1}{n}}. \tag{LLL4}$$

Proof. (LLL1) By inequality 2.9 of the previous proof, if we replace δ by 3/4 and use induction we have that

$$\|\vec{b_j^*}\|^2 \leq 2^{i-j}\|\vec{b_i^*}\|^2.$$

This inequality along with equation 1.4 gives

$$\begin{aligned}
\|\vec{b_i}\|^2 &= \|\vec{b_i^*}\|^2 + \sum_{j=1}^{i-1} \mu_{i,j}^2 \|\vec{b_j^*}\|^2 \\
&\leq \|\vec{b_i^*}\|^2 + \sum_{j=1}^{i-1} \frac{1}{4} 2^{i-j} \|\vec{b_i^*}\|^2 \\
&= (1 + \frac{1}{4}(2^i - 2)) \cdot \|\vec{b_i^*}\|^2 \\
&\leq 2^{i-1} \cdot \|\vec{b_i^*}\|^2
\end{aligned}$$

which means that

$$\|\vec{b_j}\|^2 \leq 2^{j-1} \cdot \|\vec{b_j^*}\|^2 \leq 2^{i-1} \cdot \|\vec{b_i^*}\|^2.$$

(LLL2) We can derive that one immediately by theorem 2.3.5 if we replace δ with 3/4.
(LLL3) Recall that $\|\vec{b_i^*}\| \leq \|\vec{b_i}\|$. In addition,by definition $det(\mathcal{L}(B)) = vol(\mathcal{P}(B))$ we know that

$$det(\mathcal{L}) = \prod_{i=1}^{n} \|\vec{b_i^*}\| \leq \prod_{i=1}^{n} \|\vec{b_i}\|$$

which proves the first part of (LLL3). For the second part we use the fact that $\|\vec{b_i}\| \leq 2^{(i-1)/2} \|\vec{b_i^*}\|$. Thus

$$\begin{aligned}
\prod_{i=1}^{n} \|\vec{b_i}\| \leq \prod_{i=1}^{n} 2^{(i-1)/2} \|\vec{b_i^*}\| &= \prod_{i=1}^{n} \|\vec{b_i^*}\| \cdot 2^{\sum_{i=1}^{n}(i-1)/2} \\
&= \prod_{i=1}^{n} \|\vec{b_i^*}\| \cdot 2^{\frac{n(n-1)}{4}} = 2^{\frac{n(n-1)}{4}} \cdot det(\mathcal{L}).
\end{aligned}$$

(LLL4) By (LLL1)

$$\|\vec{b_1}\| \leq 2^{\frac{i-1}{2}} \cdot \|\vec{b_i^*}\|.$$

Thus

$$\begin{aligned}
\prod_{i=1}^{n} \|\vec{b_1}\| \leq \prod_{i=1}^{n} 2^{(i-1)/2} \|\vec{b_i^*}\| &= 2^{\frac{n(n-1)}{4}} \cdot det(\mathcal{L}) \\
\Rightarrow \|\vec{b_1}\| &\leq 2^{\frac{n-1}{4}} \cdot det(\mathcal{L})^{\frac{1}{n}}.
\end{aligned}$$

□

Remark 2.3.8. All the above inequalities hold for arbitrary parameter δ if we replace 2 with $\frac{4}{4\delta-1}$.

Corollary 2.3.9. Let $\mathcal{L} \in \mathbb{Z}^n$ be a lattice (notice that we require that $\mathcal{L} \in \mathbb{Z}^n$ and not $\mathbb{R}^n$). Then, the LLL algorithm outputs a reduced basis $\{\vec{b_1}, \vec{b_2}, ..., \vec{b_n}\}$ such that

$$\|\vec{b_i}\| \leq 2^{\frac{n(n-1)}{4(n-i+1)}} \cdot det(\mathcal{L})^{\frac{1}{n-i+1}} \quad for \quad i = 1, 2, ..., n. \tag{2.10}$$

2.3.2 LLL Algorithm

The LLL Algorithm is given below (algorithm 4). During the execution of the algorithm we have to compute the vectors $\vec{b_1^*}, \vec{b_2^*}, ..., \vec{b_i^*}$. We do so by using the Gram-Schmidt Orthogonalization Algorithm (algorithm 1).

Algorithm 4: LLL Reduction Algorithm

```
**Input**: Lattice basis $\vec{b_1}, \vec{b_2}, ..., \vec{b_n} \in \mathbb{Z}^n$.
**Output**: A $\delta$-LLL-Reduced Basis for $\mathcal{L}(B)$.
Start: compute $\vec{b_1^*}, \vec{b_2^*}, ..., \vec{b_n^*}$
Reduction Step :
for $i = 2$ to $n$ do
    for $j = i - 1$ downto $1$ do
        $b_i \leftarrow b_i - c_{i,j} b_j$ where $c_{i,j} = \lceil \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \rfloor$;
    end
end
Swap Step:
if there is $i$ such that $\delta \|\vec{b_i^*}\|^2 > \|\vec{b_{i+1}^*} + \mu_{i+1,i} \cdot \vec{b_i^*}\|^2$ then
    $\vec{b_i} \leftrightarrow \vec{b_{i+1}}$;
    goto Start
end
return $\vec{b_1}, \vec{b_2}, ..., \vec{b_n}$
```

Remark 2.3.10. The above algorithm is almost the same with Gram-Schmidt Orthogonalization Algorithm (algorithm 1). Their difference lies in the fact that in the above algorithm we round the coefficients $\mu_{i,j}$ (denoted in the algorithm by $c_{i,j}$ in order to avoid confussion) to the closest integer. Since the new base B' is obtained by the initial B by a sequence of elementary column operations, B, B' are clearly bases of the same lattice.

Let us now take a deeper look in the above algorithm that will allow as to prove its correctness.

Reduction step: This step takes care of the first property of an LLL reduced basis. Throughout this step the Gram-Schmidt basis $\vec{b_1^*}, \vec{b_2^*}, ..., \vec{b_n^*}$ does not change (since we only perform column operations of the form $b_i \leftarrow b_i + ab_j$ which do not affect the Gram-Schmidt basis) and we therefore do not have to recompute the Gram-Schmidt vectors. The invariant of the outer loop in that step is that in the ith iteration, the projection of $\vec{b_i}$ on $\vec{b_j^*}$ for any $(j < i)$ is at most $\frac{1}{2}\|\vec{b_j^*}\|$. It is very important to note that the inner loop goes from $i - 1$ down to 1.

To make the above more clear we give below an instance of the current basis B during the execution of the algorithm (the example was taken from [33]). Consider the ith iteration of the outer loop and the value $j = 2$ for the inner loop. The matrix B at this point looks like:

$$\begin{bmatrix} \|\vec{b_1^*}\| & \leq \frac{1}{2}\|\vec{b_1^*}\| & \leq \frac{1}{2}\|\vec{b_1^*}\| & \cdots & * & * & \cdots \\ 0 & \|\vec{b_2^*}\| & \leq \frac{1}{2}\|\vec{b_2^*}\| & \cdots & * & * & \cdots \\ 0 & & \|\vec{b_3^*}\| & \cdots & \leq \frac{1}{2}\|\vec{b_3^*}\| & * & \cdots \\ \vdots & & \ddots & & & \vdots & \\ & & & & \leq \frac{1}{2}\|\vec{b_{i-1}^*}\| & * & \\ 0 & \cdots & & & \|\vec{b_i^*}\| & * & \cdots \\ & & & & 0 & \|\vec{b_{i+1^*}}\| & \cdots \\ \vdots & & & & \vdots & & \ddots \end{bmatrix}.$$

Notice that the first property holds for all columns coresponding to $j < i$. In addition, at this point (execution where still $j = 2$) the property holds for the ith column for all elements that belong to rows with index greater than 2. The reader here is asked to justify the necessity of counting j from $i-1$ downto 1 and not the other way.

Swap Step: This step takes care of the second property of an LLL-Reduced basis. If the algorithm terminates then the algorithm guarantees that the second property is satisfied.

The following lemma proves the correctness of the LLL algorithm.

Lemma 2.3.11 (LLL Correctness). If the LLL algorithm described above terminates, then its output is a δ-LLL-Reduced Basis for the lattice spanned by the input basis $\vec{b_1}, \vec{b_2}, ..., \vec{b_n}$.

Proof. To prove the above lemma , we have to prove that both properties of the LLL-Reduction definition are satisfied upon the algorithm's termination and that the basis produced is equivalent to the initial basis in that they both produce the same lattice. The satisfaction of the second property is enforced by the swap step of the algorithm. In addition the output basis is equivalent to the input basis since the algorithm only performs elementary column operations (notice that in every elementary operation of the form $b_i \leftarrow b_i + ab_j$, $a \in \mathbb{Z}$).It only remains to show that the first property is also satisfied upon termination. Recall first that throughout the reduction step the Gram-Schmidt basis remains unchanged. Consider now the ith iteration of the outer loop and the jth iteration of the inner loop (where $i > j$).Then immediately after this iteration we have

$$|\mu_{i,j}| = |\frac{\langle b_i - c_{i,j}b_j, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}| = |\frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} - \lceil \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \rfloor \cdot \frac{\langle b_j, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}| \leq \frac{1}{2}.$$

In the above expression the first equality follows from the definition of the reduction step while the last inequality follows from the fact that $\langle b_j, b_j^* \rangle = \langle b_j^*, b_j^* \rangle$ (Recall equation 1.4 to see that). □

2.3.3 Running Time Analysis

For the running time analysis of the LLL Algorithm we divide our proof that LLL Algorithm runs in polynomial time into two steps. In the first step we prove that the number of iterations is polynomially bounded by the input size, while in the second step we prove that the running time of each iteration is polynomially bounded too.

We start by proving that the number of iterations is polynomially bounded. In the following analysis for simplicity's sake we write b_i instead of $\vec{b_i}$ to indicate the basis vectors.

Lemma 2.3.12. For every integer basis $B \in \mathbb{Z}^{m\times n}$ we have that $det(\mathcal{L}(B))^2 \in \mathbb{Z}$.

Proof. In the standard scalar product $det(\mathcal{L})^2 = det(B^T B)$ which is clearly an integer since $B \in \mathbb{Z}^{m\times n}$. □

We can therefore associate the following integer to the basis B.

Definition 2.3.13
$\mathcal{D} = \prod_{k=1}^{n} det(\mathcal{L}(b_1, b_2, ..., b_k))^2 = (\prod_{k=1}^{n} \|b_1^*\|\|b_2^*\| \cdots \|b_k^*\|)^2$.

We also define $M = max\{n, log(\max_i \|b_i\|)\}$. This is a lower bound on the input size.We will now show that $\mathcal{D}$ decreases at least by a factor of δ at each iteration.

Lemma 2.3.14. The number of iterations in the LLL Algorithm is polynomial in M.

Proof. We have already mentioned that during the reduction step the vectors $b_1^*, b_2^*, ..., b_n^*$ remain unchanged and so does $\mathcal{D}$.

Consider now a swap step. Assume that the swap was performed between b_i and b_{i+1} and let $\mathcal{D}, \mathcal{D}'$ be the integers associated to the basis B before and after the swap respectively.Then we have

$$\begin{aligned}\frac{\mathcal{D}}{\mathcal{D}'} &= \frac{\prod_{k=1}^{n} det(\mathcal{L}(b_1, b_2, ..., b_k))^2}{\prod_{k=1}^{n} det(\mathcal{L}(b'_1, b'_2, ..., b'_k))^2} \\ &= \frac{(\prod_{k=1}^{i-1} det(\mathcal{L}(b_1, ..., b_k))^2) \cdot (det(\mathcal{L}(b_1, ..., b_i))^2) \cdot (\prod_{k=i+1}^{n} det(\mathcal{L}(b_1, ..., b_k))^2)}{(\prod_{k=1}^{i-1} det(\mathcal{L}(b'_1, ..., b'_k))^2) \cdot (det(\mathcal{L}(b'_1, ..., b'_i))^2) \cdot (\prod_{k=i+1}^{n} det(\mathcal{L}(b'_1, ..., b'_k))^2)} \\ &= \frac{det(\mathcal{L}(b_1, ..., b_i))^2}{det(\mathcal{L}(b'_1, ..., b'_i))^2}.\end{aligned}$$

The last equality can be justified as follows:
if $k < i$ then obviously $b'_l = b_l$ for $l = 1, 2, ..., k$ as the first $i-1$ vectors remain unchanged after the swap. So obviously $det(\mathcal{L}(b_1, ..., b_k))^2 = det(\mathcal{L}(b'_1, ..., b'_k))^2$.
If $k > i$ then $\mathcal{L}(b_1, ..., b_k)$ contains exactly the same vectors as $\mathcal{L}(b'_1, ..., b'_k)$. Both

lattices contain b_i, b_{i+1} in a different order but this does not affect their determinant so obviously again $det(\mathcal{L}(b_1, ..., b_k))^2 = det(\mathcal{L}(b'_1, ..., b'_k))^2$.
In addition

$$det(\mathcal{L}(b_1, ..., b_i)) = \prod_{j=1}^{i} \|b_j^*\|$$

while

$$det(\mathcal{L}(b'_1, ..., b'_i)) = \prod_{j=1}^{i} \|b_j'^*\|.$$

Recall now that $b'_l = b_l$ for $l = 1, 2, ..., i-1$ and by Gram-Schmidt process $b_l'^* = b_l^*$ for $l = 1, 2, ..., i-1$. On the other hand

$$\begin{aligned} b_i'^* &= b'_i - \sum_{j=1}^{i-1} \mu_{i',j} b_j'^* = b_{i+1} - \sum_{j=1}^{i-1} \mu_{i+1,j} b_j^* \\ &= b_{i+1} - \sum_{j=1}^{i} \mu_{i+1,j} b_j^* + \mu_{i+1,i} b_i^* = b_{i+1}^* + \mu_{i+1,i} b_i^*. \end{aligned}$$

Thus

$$\begin{aligned} \frac{\mathcal{D}}{\mathcal{D}'} &= \frac{det(\mathcal{L}(b_1, ..., b_i))^2}{det(\mathcal{L}(b'_1, ..., b'_i))^2} = \frac{\prod_{j=1}^{i} \|b_j^*\|^2}{\prod_{j=1}^{i} \|b_j'^*\|^2} \\ &= \frac{\|b_i^*\|^2}{\|b_i'^*\|^2} = \frac{\|b_i^*\|^2}{\|b_{i+1}^* + \mu_{i+1,i} b_i^*\|^2} > \frac{1}{\delta}. \end{aligned}$$

The last inequality is dictated by the swap condition. If it didn't hold then there would be no swap. Suppose that the initial value of $\mathcal{D}$ is D and that the algorithm terminates after m iterations (the fact that the integer $\mathcal{D}$ decreases at each step while remaining integer, guarantees the termination of the algorithm). The above inequality gives ($\mathcal{D}^{(i)}$ denotes the value of $\mathcal{D}$ after the ith itearation)

$$D > \frac{1}{\delta}\mathcal{D}^{(1)} \Rightarrow D > (\frac{1}{\delta})^m \mathcal{D}^{(m)} > (\frac{1}{\delta})^m$$

since $\mathcal{D}$ is a nonzero integer. Thus

$$(\frac{1}{\delta})^m < D \Rightarrow m < \log_{\frac{1}{\delta}} D.$$

Now in order to bound D we recall that $\|b_i^*\| \leq \|b_i\|$. This gives

$$D = \prod_{k=1}^{n} (\|b_1^*\|\|b_2^*\| \cdots \|b_k^*\|)^2 \leq \prod_{k=1}^{n} (\|b_1\|\|b_2\| \cdots \|b_k\|)^2 \leq \max_i \|b_i\|^{n(n+1)}.$$

This finally gives

$$m < \log_{\frac{1}{\delta}} D \leq \log_{\frac{1}{\delta}} \max_i \|b_i\|^{n(n+1)} = n(n+1) \log_{\frac{1}{\delta}} \max_i \|b_i\| \leq n(n+1)M$$

which completes the proof as M is a lower bound for the input size. □

We still need to show that that each iteration also takes polynomial time. Apparently the number of arithmetic operations performed at each iteration is polynomial. Thus, it only remains to show that the numbers involved in the computations can be represented using a polynomial number of bits. We denote $\mathcal{D}_i = det(\mathcal{L}(b_1, b_2, ..., b_i))^2 = det(B_i)^2$.Thus

$$\mathcal{D} = \prod_{k=1}^{n} \mathcal{D}_i.$$

We first prove the following lemma.

Lemma 2.3.15. The following statements are true for the LLL algorithm:

(a) The Gram-Schmidt vectors $b_1^*, b_2^* ..., b_n^*$ can be computed in polynomial time in M.

(b) $\mathcal{D} b_i^* \in \mathbb{Z}^n$.

(c) $\|b_i^*\| \leq \mathcal{D}$ for every $i = 1, 2, ..., n$.

Proof. (a)Consider the equation 1.4. By induction $b_i^* - b_i \in span(b_1, b_2, ..., b_{i-1})$. We can therefore write $b_i^* = b_i + \sum_{j=1}^{i-1} a_j b_j$ for some $a_1, a_2, ..., a_{i-1}$. We will now show that we can compute the coefficients a_i in polynomial time. Recall that $b_l \in span(b_1^*, b_2^*, ..., b_l^*)$ which implies that $\langle b_i^*, b_l \rangle = 0$ for every $l = 1, 2, ..., i-1$.This gives:

$$\langle b_i^*, b_l \rangle = \langle b_i + \sum_{j=1}^{i-1} a_j b_j, b_l \rangle = \langle b_i, b_l \rangle + a_1 \langle b_1, b_l \rangle + a_2 \langle b_2, b_l \rangle + ... + a_{i-1} \langle b_{i-1}, b_l \rangle = 0$$

If we now consider all the vectors b_l, $l = 1, 2, ..., i-1$, we obtain the following system of $i-1$ linear equations in $i-1$ variables:

$$\begin{aligned} a_1 \langle b_1, b_1 \rangle + a_2 \langle b_2, b_1 \rangle + ... + a_{i-1} \langle b_{i-1}, b_1 \rangle &= -\langle b_i, b_1 \rangle \\ a_1 \langle b_1, b_2 \rangle + a_2 \langle b_2, b_2 \rangle + ... + a_{i-1} \langle b_{i-1}, b_2 \rangle &= -\langle b_i, b_2 \rangle \\ &\vdots \\ a_1 \langle b_1, b_{i-1} \rangle + a_2 \langle b_2, b_{i-1} \rangle + ... + a_{i-1} \langle b_{i-1}, b_{i-1} \rangle &= -\langle b_i, b_{i-1} \rangle. \end{aligned}$$

We can now find the coefficients $a_1, a_2, ..., a_{i-1}$ in polynomial time by solving the above system using Cramer's rule.We can then compute the vectors b_i^* through the equation $b_i^* = b_i + \sum_{j=1}^{i-1} a_j b_j$.
(b) The solution for each a_j can be written in the following form:

$$a_j = \frac{det(\text{some integer matrix})}{det \begin{pmatrix} \langle b_1, b_1 \rangle & \cdots & \langle b_{i-1}, b_1 \rangle \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \langle b_1, b_{i-1} \rangle & \cdots & \langle b_{i-1}, b_{i-1} \rangle \end{pmatrix}} = \frac{\alpha \in \mathbb{Z}}{det B_{i-1}^T B_{i-1}} = \frac{\alpha \in \mathbb{Z}}{\mathcal{D}_{i-1}}.$$

This combined with the fact that $b_i^* = b_i + \sum_{j=1}^{i-1} a_j b_j$ clearly implies that $\mathcal{D}_{i-1} b_i^* \in \mathbb{Z}^n$ and since $\mathcal{D}_{i-1}$ divides $\mathcal{D}$ by definition, we finally get that $\mathcal{D} b_i^* \in \mathbb{Z}^n$.
(c) By definition $\mathcal{D}_i = (\prod_{j=1}^{i-1} \|b_j^*\|^2) \cdot \|b_i^*\|^2$. Thus

$$\|b_i^*\|^2 = \frac{\mathcal{D}_i}{\prod_{j=1}^{i-1} \|b_j^*\|^2} \leq \mathcal{D}_i \cdot \prod_{j=1}^{i-1} \mathcal{D}_j^2 \leq \mathcal{D}^2 \Rightarrow \|b_i^*\| \leq \mathcal{D}$$

where the first inequality follows from the fact that $\mathcal{D} b_i^* \in \mathbb{Z}^n$ and thus $\|\mathcal{D}_i \cdot b_i^*\|^2 \geq 1 \Rightarrow \frac{1}{\|b_i^*\|^2} \leq \mathcal{D}_i^2$. □

In order to complete the running time analysis we still need to show that the vectors b_i do not grow too large during the execution of LLL algorithm.Notice that during the reduction step, vectors b_i do change so we need to prove an upper bound on their norm. This is proved in the following lemma.

Lemma 2.3.16. All vectors b_i appearing during an iteration can be represented using $poly(M)$ bits.

Proof. Equation 1.4 says that $b_i = b_i^* + \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$. In addition, $|\mu_{i,j}| \leq \frac{1}{2}$ and $\langle b_i^*, b_j^* \rangle = 0$ for each $i \neq j$. Thus

$$\|b_i\|^2 = \|b_i^*\|^2 + \sum_{j=1}^{i-1} \mu_{i,j}^2 \|b_j^*\|^2 \leq \mathcal{D}^2 + \frac{n}{4}\mathcal{D}^2 \leq n\mathcal{D}^2.$$

Remember now that by hypothesis, in the beggining of the algorithm $b_i \in \mathbb{Z}^n$ for each i and that we only perform elementary integer column operations. Thus b_i's remain integer throughout the algorithm.Since their norm is bounded too, then they can be represented with poly(M) bits.In addition

$$|c_{i,j}| = |\lceil \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \rfloor| \leq \frac{\|b_i\| \cdot \|b_j^*\|}{\|b_j^*\|^2} + 1 = \frac{\|b_i\|}{\|b_j^*\|} + 1 \leq \frac{\|b_i\|}{1/\mathcal{D}} + 1 \leq 2\mathcal{D}\|b_i\|.$$

(We have used Cauchy-Schwartz Inequality for the first inequality.) Using the above we finally obtain:

$$\begin{aligned} \|b_i - c_{i,j} b_j\| &\leq \|b_i\| + |c_{i,j}| \|b_j\| \\ &\leq (1 + 2\mathcal{D}\|b_j\|)\|b_i\| \\ &\leq (1 + 2\mathcal{D}\sqrt{n}\mathcal{D})\|b_i\| \\ &\leq (4n\mathcal{D})^2 \|b_i\| \end{aligned}$$

which is obviously representable in $poly(M)$ bits. □

The following theorem recapitulates the facts that we have proved so far.

Theorem 2.3.17 (**LLL Running Time**)
The running time of the LLL Algorithm is polynomial in its input size.

Proof. The proof is immediate by inspection of the algorithm 4 and the lemmas 2.3.14, 2.3.15 and 2.3.16. □

2.3.4 Finding Solutions to the Simultaneous Diophantine Approximation Problem

In order to give a first flavor of LLL applications, we demonstrate below a use of the LLL algorithm in finding a solution to the Simultaneous Diophantine Approximation Problem (SDAP). This was one of the first applications of LLL algorithm presented in the LLL paper (see [24] for more details). The following approach is followed (in a similar fashion) in a large number of applications throughout the rest of the thesis. The SDAP as defined in [19] is the following:

Definition 2.3.18 (**Simultaneous Diophantine Approximation Problem**)
Given $a_1, a_2, ..., a_n \in \mathbb{Q}, \epsilon > 0 \in \mathbb{Q}, Q > 0$, find integers $p_1, p_2, ..., p_n$ and q such that $0 < q \leq Q$ and

$$\left| a_i - \frac{p_i}{q} \right| \leq \frac{\epsilon}{q} \qquad i = 1, 2, ..., n.$$

Theorem 2.3.19
There exists a polynomial time algorithm that, given a positive integer n and rational numbers $a_1, a_2, ..., a_n, \epsilon$ satisfying $0 < \epsilon < 1$, finds integers $p_1, p_2, ..., p_n, q$ for which

$$|p_i - qa_i| \leq \epsilon \quad for \quad 1 \leq i \leq n,$$
$$1 \leq q \leq 2^{n(n+1)/4}\epsilon^{-n}.$$

Proof. Consider the lattice $\mathcal{L}$ of rank $n+1$ spanned by the columns of the following $(n+1) \times (n+1)$-matrix

$$B = \begin{bmatrix} 1 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & \cdots & 0 & -a_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -a_n \\ 0 & 0 & \cdots & 0 & 2^{-n(n+1)/4}\epsilon^{n+1} \end{bmatrix}.$$

Then theorem 2.3.7 says that that there exists a polynomial-time algorithm which finds a reduced basis $b_1, b_2, ..., b_{n+1}$ for $\mathcal{L}$ such that

$$\|b_1\| \leq 2^{\frac{n+1-1}{4}} \cdot det(\mathcal{L})^{\frac{1}{n+1}} = \epsilon.$$

In addition $b_1 \in \mathcal{L}$ that is there exists a vector $\vec{x} = (p_1, p_2, ..., p_n, q)^T$ with integer components such that

$$b_1 = B\vec{x} = (p_1 - qa_1, p_2 - qa_2, ..., p_n - qa_n, q \cdot 2^{-n(n+1)/4} \epsilon^{n+1})^T.$$

Thus $\|b_1\| \leq \epsilon$ implies that

$$\begin{array}{c} |p_i - qa_i| \leq \epsilon \quad for \quad 1 \leq i \leq n, \\ |q| \leq 2^{n(n+1)/4} \epsilon^{-n} \end{array} .$$

In addition , the requirement $\epsilon < 1$ implies that $q \neq 0$ (otherwise $\|b\| \geq \min_i |p_1| \geq 1$). In addition we can replace (if needed) b_1 with $-b_1$ to obtain $q > 0$. □

2.3.5 LLL Applications

The LLL Algorithm also has plenty of applications in various fields of computer science. We briefly give a description of some of them here. In the following chapters we will present in detail some of the LLL applications related to cryptography.

1. Factoring Polynomials over the Integers or the rational numbers. This was the initial use of the LLL algorithm in paper [24].

2. Finding the minimal polynomial (with integer coefficients) of an algebraic number given to a good enough approximation. For instance, the minimal integer polynomial, a root of whose approximates "sufficiently" the number 1.7321, is $x^2 - 3$.

3. Integer Programming. While this problem is NP-Complete in its general setting, one can use LLL to obtain a polynomial time solution to an Integer Programming Problem with a fixed number of variables.

4. Approximation of the Closest Vector Problem,that is the Problem of finding the lattice point that is closest to a given point (which does not necessarily belong to the lattice).

5. Many applications both in Cryptanalysis and in establishing Cryptographic Primitives. The next chapters of the thesis are devoted to that kind of applications.

Chapter 3

Finding Small Roots to Polynomial Equations

3.1 Introduction

In the previous two chapters, we presented the basic definitons and properties of lattices. We also introduced the notion for reduced basis and presented in detail LLL algorithm, a polynomial time algorithm that produces basis vectors which approximate the shortest vectors in a lattice. Some of these results will be used throughout this chapter.

We summarize some of the important results concerning LLL in the following theorem for convenience. We also restate the results obtained by Minkowski's Convex Body Theorem.

Theorem 3.1.1 (**LLL Results**)
Let $[\vec{b_1}, \vec{b_2}, ..., \vec{b_n}]$ be a basis for the lattice $\mathcal{L} \in \mathbb{R}^n$ and $\vec{b_1}^*, \vec{b_2}^*, ..., \vec{b_n}^*$ the corresponding Gram-Schmidt vectors. Then LLL algorithm produces in polynomial time an equivalent reduced basis $[\vec{b_1}', \vec{b_2}', ..., \vec{b_n}']$ the vectors of which satisfy the following inequalities:

$$\|\vec{b_j}'\|^2 \leq 2^{i-1} \cdot \|\vec{b_1^*}\|^2 \qquad for\, 1 \leq j \leq i \leq n, \tag{LLL1}$$

$$\|\vec{b_1}'\| \leq 2^{\frac{n-1}{2}} \lambda_1(\mathcal{L}) \tag{LLL2}$$

$$det(\mathcal{L}) \leq \prod_{i=1}^{n} \|\vec{b_i}'\| \leq 2^{\frac{n(n-1)}{4}} \cdot det(\mathcal{L}) \tag{LLL3}$$

$$\|\vec{b_1}'\| \leq 2^{\frac{n-1}{4}} \cdot det(\mathcal{L})^{\frac{1}{n}} \tag{LLL4}$$

If in addition $\mathcal{L} \in \mathbb{Z}^n$ (and not generally $\mathbb{R}^n$), then the LLL algorithm outputs a reduced basis $[\vec{b_1}', \vec{b_2}', ..., \vec{b_n}']$ such that

$$\|\vec{b_i}\| \leq 2^{\frac{n(n-1)}{4(n-i+1)}} \cdot det(\mathcal{L})^{\frac{1}{n-i+1}} \quad for \quad i = 1, 2, ..., n. \tag{3.1}$$

Theorem 3.1.2 (**Convex Body Theorem and its Results**)
Consider a full dimensional lattice ($n = m$). Then:

$$SVP_\infty \leq \sqrt[n]{det(\mathcal{L})} \text{ and} \tag{SVI1}$$
$$SVP_2 < \sqrt{n}\,\sqrt[n]{det(\mathcal{L})} \tag{SVI2}$$

where SVP_i denotes the Shortest Vector in the lattice with respect to norm l_i. We emphasize that the above theorem is existensial in that it only proves the existence but not a way to find such a vector.

In the current chapter we present some recently proposed applications of lattices in finding small roots to polynomial equations. In particular, in section 3.2 we present how one can use lattice theory to solve modular polynomial equations. We present in detail the corresponding technique for univariate modular equations and outline the extension to more than one variables. In section 3.3 we describe a lattice-based approach for finding small roots to polynomial equations over the integers (and not only modulo a number). We mainly focus on the bivariate case and give the underlying ideas for extending the approach to multivariate polynomials.

While this chapter does not include any pure cryptographic applications, it provides some very useful theorems that will make the analysis of the following two chapters much easier to follow. Of course, the results presented here are of independent interest and can be applied in other fields as well.

3.2 Modular Polynomial Equations

In this section we present some methods for finding small roots to modular polynomial equations.We emphasize here that we are only interested in integer roots. Such methods have found a large number of applications in Cryptography. Some of these applications will be presented in detail in the next two chapters.

3.2.1 Univariate Case

We first set the goal of this subsection. Let N be some large integer of unknown factorization and $f \in \mathbb{Z}[x]$ be a polynomial of degree d. Consider also the following univariate modular equation:

$$f(x) = a_d x^d + a_{d-1} x^{d-1} + ... + a_1 x + a_0 \equiv 0\,(mod\,N). \tag{3.2}$$

In general there is no known efficient algorithm that finds integer roots of the above equation. However, Coppersmith [10, 11, 12] introduced an efficient method for finding small integer solutions using the LLL algorithm.

To illustrate a simple example of how such an approach can work, consider the case where we want to find a root to the following modular equation $f(x) = x^d - c \equiv$

$0\,(mod\,N)\quad x, c \geq 0$. Assume now that we are given the additional information that there exists a solution x_0 such that $x_0 \leq N^{\frac{1}{d}}$. That would imply that

$$f(x_0) \equiv 0\,(mod\,N), \text{ and } \quad |f(x_0)| = |(x_0)^d - c| < N.$$

The above two conditions give that $f(x_0) = 0$ over $\mathbb{Z}$ and thus we can recover x_0 using an ordinary root finding algorithm. Note that we are searching for $x_0 \in \mathbb{Z}$ so the recovery of such a x_0 is not hard.

We would like now to generalize the above technique for arbitrary polynomials and improve the bounds on the solutions we can recover. The following analysis presents the various advancements in a chronological order. It is important for the reader to understand the steps followed till the final results as well as the basic underlying ideas since variants of this technique will be used in solving multivariate modular equations and in finding small roots to multivariate integer equations. We will therefore provide a detailed presentation of the technique. We first give some notation and definitions.

Notation and Definitions

Let $f(x) := \sum_i a_i x^i$ be univariate polynomial with coefficients in $a_i \in \mathbb{Z}$. All terms x^i with nonzero coefficients are called *monomials*. We will frequently represent a polynomial with the respective vector of its coefficients. For example the polynomial $p(x) = 3x^3 + 2x + 20$ will be represented by the vector $p = (20, 2, 0, 3)$. This notation will prove to be very useful in the next sections. Finally we define the norm of a polynomial f as the Euclidean norm of its coefficient vector:

$$\|f\|^2 := \sum_i a_i^2.$$

Definition 3.2.1 (**Root Equivalent polynomials**)
Consider two polynomials f, g. We say that f is a **root container** of g if each root of g is also a root of f. When the roots are considered modulo N, we say that f is a root container of g modulo N.

In the following analysis, the notation $\|\cdot\|$ will always imply the Euclidean norm, that is $\|\cdot\|_2$. In case we want to use another norm, we will explicitly do so by putting the respective index to the norm notation.

Key Ideas

Suppose that we are given a polynomial

$$f(x) = a_d x^d + a_{d-1} x^{d-1} + ... + a_1 x + a_0$$

and we are asked to find a solution to the unimodular equation

$$f(x) \equiv 0\,(mod\,N).$$

Since there are no known techniques for this general case, there is not much to do. However, if we somehow knew that $|f(x)| < N$ for all x such that $|x| \leq X$ for a certain bound X, we could easily find roots x_0 by simply solving $f(x_0) = 0$ over $\mathbb{Z}$ using a standard root finding algorithm.This seems too simple to be efficient. Indeed, there are two major drawbacks of this technique:

1. How can we actually know that such a bound X exists and how can we estimate the value of this bound? More importantly, how can we check the inequation $|f(x)| < N$ for an arbitrary polynomial $f(x)$ and define the bound X for which this inequality holds ?

2. How possible is the satisfaction of the above inequality and even if it is satisfied for a bound X is that bound "large" enough to recover a significant percentage of the solutions?

We will now try to handle both inconveniences.

1. Instead of searching for a bound X such that $|f(x_0)| < N$ for every modular solution x_0 with $|x_0| \leq X$, we can use a stricter condition. Suppose for example that we can find a bound X such that $\sum_{i=0}^{d} |f_i x_0^i| < N$. This condition is stricter in that it is satisfied by at most all the polynomials that satisfy the initial condition $|f(x_0)| < N$ for all x_0 such that $|x_0| \leq X$. Indeed, the condition $\sum_{i=0}^{d} |f_i x_0^i| < N$ implies that

$$|f(x_0)| = |\sum_{i=1}^{d} f_i x_0^i| \leq \sum_{i=0}^{d} |f_i x_0^i| < N$$

   where the first inequality stems from the extended triangle inequality.
   But $\sum_{i=0}^{d} |f_i x_0^i| < N$ is still difficult to test. We can further replace it with the even stricter condition

$$|f_i x_0^i| < \frac{N}{d+1} \text{ for all } i = 0, 1, ..., d.$$

   This can in turn be replaced by the stricter condition

$$\max_{0 \leq i \leq d} |f_i x_0^i| < \frac{N}{d+1}.$$

   For the need of this thesis we will mainly use the euclidean norm. The following theorem gives a sufficient condition the euclidean norm of the coefficient vector $\|f(xX)\|$ should satisfy in order to make the transformation of a modular equation to an analogous (in terms of "small solutions") equation over the integers possible.

   Lemma 3.2.2 (Howgrave-Graham for Univariate Polynomials). Let $h(x) \in \mathbb{Z}[x]$ be a univariate polynomial with at most ω monomials. Suppose in addition that h satisfies the following two conditions:

(a) $h(x_0) \equiv 0 (mod\, N)$ where $|x_0| < X$ and

(b) $\|h(xX)\| \leq N/\sqrt{\omega}$.

Then $h(x_0) = 0$ holds over the integers.

Proof.

$$\begin{aligned}|h(x_0)| = |\sum_i h_i x_0^i| \leq \sum_i |h_i x_0^i| = \sum_i |h_i| \left|\frac{x_0}{X}\right|^i X^i \\ \leq \sum_i |h_i| X^i \leq \sqrt{\omega} \|h(xX)\| < N.\end{aligned}$$

The last but one inequality is a direct use of Cauchy-Schwarz inequality. □

The above lemma gives a condition that may not be that tight (there may be polynomials h that do not satisfy the conditions of the lemma, but still satsify $|h(x_0)| < N$) but it is easily testable and will thus be used in the rest of the analysis.

2. As far as the second inconvenience is concerned, it is clear that in order to apply the above lemma we have to find a bound X such that $\|f(xX)\| \leq N/\sqrt{\omega}$. But how "large" such a bound can be? Can we do anything to push the bound X to values that are sufficiently large? The answer, interestingly, is "YES". The key idea is that instead of trying to find solutions to the polynomial f, we can construct new polynomials g which are root containers of f and in addition, satisfy $\|g(xX)\| \leq N/\sqrt{\omega}$ for bounds X that are significantly larger than the bounds obtained by the corresponding condition for f. In the following paragraph we present gradually the construction of such polynomials and prove the bounds achieved for each construction.

Early Constructions

For the rest of the analysis we will assume wlog that f is monic, that is the coefficient of x^d is 1. We can always transform f to such a form by multiplying it with f_d^{-1} mod N (if $(f_d, N) \neq 1$ then we have found a non trivial factor of N which significantly simplifies things). In addition, the basis vectors of a lattice will occasionally be thought as row vectors instead of column vectors. Since we are only interested in the determinant of the lattice used , this modification does not affect the subsequent analysis at all.
In order to construct a polynomial g that is a root container of f, we first consider the following set of polynomials:

$$\mathcal{Z}_1 = \{N, Nx, Nx^2, ..., Nx^{d-1}, f(x)\}.$$

It is important to notice here that any integer combination of these polynomials has at least all the roots of f modulo N. That is, if x_0 satisfies $f(x_0) \equiv 0$ modulo N, then $g(x_0) \equiv 0$ modulo N for every g that is a linear combination of the polynomials in $\mathcal{Z}_1$. Hence, it suffices to find an integer combination of those polynomials that satisfies property (b) in lemma 3.2.2. Here is the point where lattices and LLL algorithm come to the play. Consider the following lattice whose columns are the coefficient vectors of the polynomials in $\mathcal{Z}_1$.

$$\mathcal{L}_1 = \begin{bmatrix} N & 0 & \cdots & 0 & f_0 \\ 0 & XN & \ddots & 0 & Xf_1 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & X^{d-1}N & X^{d-1}f_{d-1} \\ 0 & \cdots & \cdots & 0 & X^d \end{bmatrix}_{(d+1)\times(d+1)}.$$

Notice that the ith row $i = 0, 1, ..., d$ corresponds to the coefficient of x^i multiplied by X^i.Indeed, consider an arbitrary integer linear combination $\vec{c} = [c_0, c_1, ..., c_d]^T$ of the column vectors of the above matrix.Then we have that

$$\mathcal{L}_1\vec{c} = c_0 \cdot \begin{bmatrix} N \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + c_1 \cdot \begin{bmatrix} 0 \\ XN \\ 0 \\ \vdots \\ 0 \end{bmatrix} + ... + c_d \cdot \begin{bmatrix} f_0 \\ Xf_1 \\ \vdots \\ X^{d-1}f_{d-1} \\ X^d \end{bmatrix}$$
$$= [c_0N + c_df_0, c_1XN + c_df_1X, ..., c_dX^d]^T.$$

The last vector corresponds to the coefficient vector of the polynomial

$$h(x) = (c_0N + c_df_0) + (c_1N + c_df_1)x + ... + c_dx^d.$$

Clearly h is a root container of f modulo N. The order of the column vectors in matrix $\mathcal{L}_1$ is chosen in such a way so that the resulting matrix is upper triangular. We will now use the results of LLL algorithm to determine the upper bound for X. By theorem 3.1.1 we can find in polynomial time a vector $b \in \mathcal{L}_1$ such that

$$\|b\| \leq 2^{\frac{n-1}{4}} \cdot det(\mathcal{L}_1)^{\frac{1}{n}}.$$

But this vector is,by construction of the lattice, the coefficient vector of a polynomial $h(xX)$. Using the above inequality we know that LLL returns a vector the corresponding polynomial of which satisfies:

$$\|h(xX)\| \leq 2^{\frac{d+1-1}{4}} \cdot det(\mathcal{L}_1)^{\frac{1}{d+1}}.$$

In order to apply lemma 3.2.2 on $h(x)$ and recover x_0 such that $h(x_0) = 0$ over the integers, we need to solve

$$\|h(xX)\| \leq \frac{N}{\sqrt{d+1}}$$

with respect to X. Combining the above two inequalities, a sufficient condition is

$$2^{\frac{d+1-1}{4}} \cdot det(\mathcal{L}_1)^{\frac{1}{d+1}} < \frac{N}{\sqrt{d+1}}.$$

In order to proceed we now need to compute the determinant of $\mathcal{L}_1$. This computation is straightforward since $\mathcal{L}_1$ is upper triangular and thus the determinant is simply the product of the diagonal elements of $\mathcal{L}_1$. It is not difficult to see that $det(\mathcal{L}_1) = N^d X^{\frac{d(d+1)}{2}}$ which finally gives that

$$2^{\frac{d}{4}} \cdot (N^d X^{\frac{d(d+1)}{2}})^{\frac{1}{d+1}} < \frac{N}{\sqrt{d+1}} \Rightarrow X \leq k(d) N^{\frac{2}{d(d+1)}}$$

where $k(d)$ is a small enough constant that depends only on d.

Let us summarize what we have achieved so far. We have proved that given a univariate modular equation $f(x) \equiv 0(mod\, N)$ we can find in polynomial time all the roots x_0 such that $f(x_0) \equiv 0(mod\, N)$ and $|x_0| \leq k(d) N^{\frac{2}{d(d+1)}}$.The method described above illustrates the basic underlying idea.

The question now is "Can we do any better?",that is can we obtain a larger bound for X?The answer is yes and the main idea lies in the lattice that produces the root container polynomials. To see that consider the extended set of polynomials

$$\mathcal{Z}_2 = \{N, Nx, Nx^2, ..., Nx^{d-1}\} \bigcup \{f(x), xf(x), ..., x^{d-1}f(x)\}.$$

Notice that for any integer combination g of these polynomials, the roots of g contain all the roots of f. Consider now the lattice whose columns are the coefficient vectors of the above polynomials.

$$\mathcal{L}_2 = \begin{bmatrix} N & 0 & 0 & 0 & f_0 & 0 & \cdots & 0 \\ 0 & XN & \ddots & \vdots & \vdots & Xf_0 & \ddots & \vdots \\ \vdots & 0 & \ddots & 0 & \vdots & \vdots & \ddots & 0 \\ \vdots & \vdots & \ddots & X^{d-1}N & X^{d-1}f_{d-1} & \vdots & \vdots & X^{d-1}f_0 \\ \vdots & \vdots & \ddots & 0 & X^d & X^d f_{d-1} & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & X^{d+1} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & X^{2d-2}f_{d-1} \\ 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & X^{2d-1} \end{bmatrix}_{(2d)\times(2d)}$$

Using arguments exactly similar to those used for the basis matrix $\mathcal{L}_1$ the sufficient condition in order to apply lemma 3.2.2 becomes

$$2^{\frac{2d-1}{4}} \cdot det(\mathcal{L}_2)^{\frac{1}{2d}} < \frac{N}{\sqrt{2d}}.$$

where $det(\mathcal{L}_2) = N^d X^{\frac{2d(2d-1)}{2}}$. It is not difficult to see that the above condition gives

$$X \leq l(d) N^{\frac{1}{2d-1}}$$

where here again $l(d)$ is a "small" enough constant that depends only on d. The column vectors added to $\mathcal{L}_1$ to form $\mathcal{L}_2$ have resulted to a significantly larger bound X for the range of small solutions we can find.

Remark 3.2.3. It is important to note here that the polynomials $h(x)$ constructed in either case as a linear combination of the polynomials in $\mathcal{Z}$ in order to apply lemma 3.2.2, may have more roots in $\mathbb{Z}$ than $f(x)$ modulo N. However the construction guarantees that if $f(x)$ has a "small" root x_0 modulo N, then this is certainly a root over all the integers for $h(x)$. The inverse is not always true. In order to determine the "small" roots of the modular equation $f(x) \equiv 0 (mod N)$, we first have to solve the equation $h(x) = 0$ over the integers and then check which of those roots also satify $f(x) \equiv 0(mod N)$.

Remark 3.2.4. The above methods also give answer to the following purely mathematic question:how many roots modulo N can a polynomial of degree d have in the range $x \in \{-X, ..., 0, ..., X\}$? Since the number of roots of h over the integers is at least as large as the number of "small" roots of f modulo N and h has at most d integer roots (where d is the degree of the polynomial h), the above methods give an upper bound on the number of "small" roots of f modulo N.Our first approach (where we used only polynomials from the set $\mathcal{Z}_1$) says that there are at most d integer roots x_0 such that $|x_0| < c_1(d) N^{\frac{2}{d(d+1)}}$ while the second approach (where we used polynomials from the set $\mathcal{Z}_2$) says that there are at most $2d-1$ integer roots x_0 (notice that $2d$ is the dimension of lattice $\mathcal{L}_2$) such that $|x_0| < c_2(d) N^{\frac{1}{2d-1}}$.

Coppersmith's Contribution

In this paragraph we discuss further improvements to the exponent of the bound X. The main advancement over the previous results came in 1996, when Coppersmith [10, 9] increased the bound to $N^{\frac{1}{d}}$. Coppersmith managed to prove a larger bound by incorporating the following two key ideas:

1. He further enriched the set of polynomials $\mathcal{Z}$ increasing at the same time the dimension of the lattice that produces polynomials g that are root containers of f. In particular, he used the following set of polynomials:

$$\mathcal{Z}_h = \{N^{h-j-1} f(x)^j x^i | 0 \leq i < d, 0 \leq j < h\}.$$

2. He considered linear integer combinations of the above vectors modulo N^{h-1} instead of modulo N. Notice that the construction of $\mathcal{Z}_h$ is such that for any x with $f(x) \equiv 0(mod N)$, any integer combination g of these polynomials satisfies $g(x) \equiv 0(mod N^{h-1})$.

Remark 3.2.5. The analysis presented above does not exactly follow the Coppersmith's initial presentation [10, 9].Initially , Coppersmith was working with an unnatural space and thus his presentation was difficult both to follow and to be transfered to practical implementations. Coppersmith did not express the conditions that lead to the bound X in terms of polynomial arithmetic. The convenient formulation presented in this section and followed in the rest of this thesis is due to Howgrave-Graham [23] who revisited and simplified the analysis of Coppersmith's method in 1997. In fact, all current uses of Coppersmith's univariate modular method use Howgrave-Graham's approach.In this thesis we will refer to Coppersmith by keeping in mind that we actually use Howgrave-Graham's approach which is based on Coppersmith's underlying idea.

We will now prove Coppersmith's main result for univariate polynomial modular equations. In fact, we will present a generalization of Coppersmith's theorem given by May [28] in 2004 as well as the detailed proof (found in [26] (p.34-37) or in the full version of [28]). We first need the following lemma.

Lemma 3.2.6 (Generalized Howgrave-Graham for Univariate Modular Polynomials). Let $f(x) \in \mathbb{Z}[x]$ be a univariate polynomial with at most ω monomials.Further let m be a positive integer. Suppose that

1. $f(x_0) \equiv 0 (mod\, b^m)$ where $|x_0| < X$
2. $\|f(xX)\| \leq \frac{b^m}{\sqrt{\omega}}$.

Then $f(x_0) = 0$ holds over the integers.

Proof. The proof is completely analogous to the proof of lemma 3.2.2 and is therefore omitted. □

Theorem 3.2.7 (Coppersmith Generalized Theorem for Univariate Modular Equations)
Let N be an integer of unknown factorization, which has a divisor $b \geq N^{\beta}$. Furthermore, let $f(x)$ be a univariate, monic polynomial of degree δ. Then we can find all solutions x_0 for the equation $f(x) \equiv 0 (mod\, b)$ with

$$|x_0| \leq \frac{1}{2} N^{\frac{\beta^2}{\delta} - \epsilon}$$

in time polynomial in $(\log N, \delta, \frac{1}{\epsilon})$.

Proof. We first define the bound $X := \frac{1}{2} N^{\frac{\beta^2}{\delta} - \epsilon}$. We apply Coppersmith's approach as presented above. We first need to build the lattice. We fix a number m such that

$$m \geq max \left\{ \frac{\beta^2}{\delta\epsilon}, \frac{7\beta}{\delta} \right\}. \tag{3.3}$$

We then choose the set $\mathcal{Z}$ of polynomials that will form the basis of our lattice. We include the following polynomials in $\mathcal{Z}$

$$\begin{array}{lllll}
N^m, & xN^m, & x^2N^m, & \cdots & x^{\delta-1}N^m, \\
N^{m-1}f, & xN^{m-1}f, & x^2N^{m-1}f, & \cdots & x^{\delta-1}N^{m-1}f, \\
N^{m-2}f^2, & xN^{m-2}f^2, & x^2N^{m-2}f^2, & \cdots & x^{\delta-1}N^{m-2}f^2, \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
Nf^{m-1}, & xNf^{m-1}, & x^2Nf^{m-1}, & \cdots & x^{\delta-1}Nf^{m-1}.
\end{array}$$

The above polynomials are δm in total with increasing degrees (from 0 to $\delta m - 1$). We also include the following polynomials

$$f^m, \quad xf^m, \quad x^2f^m, \quad \cdots, \quad x^{t-1}Nf^m.$$

t is a parameter to be optimized as a function of m. We write the above polynomials in the following more compact form.

$$\begin{array}{ll}
g_{i,j}(x) = x^jN^if^{m-i}(x) & for\, i = 0, ..., m-1,\ j = 0, ..., \delta - 1 \\
h_i(x) = x^if^m(x) & for\, i = 0, ..., t-1.
\end{array}$$

We now construct the lattice $\mathcal{L}$ the rows of which are the coefficient vectors of $g_{i,j}(xX)$ and $h_i(xX)$. Note first that, unlike the previous constructions, the coefficient vectors form the rows of the lattice instead of the columns.This should not cause any confusion since we are only interested in the determinant of the lattice as we have already mentioned. (A description using columns would not affect the method at all. The basis of the matrix would be the transpose matrix of the matrix given below.We use rows only to be accordant with the established bibliography).Second,by a simple inspection of the polynomials, it is easy to observe that we can order the polynomials $g_{i,j}(xX)$ and $h_i(xX)$ in strictly increasing order of their degrees k. Hence we can write the basis B of the lattice $\mathcal{L}$ as a lower triangular matrix. The dimension of the lattice will be $\omega = \delta m + t$. The basis B can be then written as the following $(\omega \times \omega)$ matrix.

$$\left(\begin{array}{ccccccccccccc}
N^m &&&&&&&&&&&& \\
& N^mX &&&&&&&&&&& \\
&& \ddots &&&&&&&&&& \\
&&& N^mX^{\delta-1} &&&&&&&&& \\
\hline
& \ddots & \ddots & \ddots & \ddots &&&&&&&& \\
\hline
? & ? & ? & \cdots & ? & NX^{\delta m-\delta} &&&&&&& \\
& ? & ? & ? & \cdots & ? & NX^{\delta m-\delta+1} &&&&&& \\
&& \ddots & \ddots & \ddots & \ddots & \ddots & \ddots &&&&& \\
&&& ? & ? & ? & \cdots & ? & NX^{\delta m-1} &&&& \\
\hline\hline
? & ? & ? & ? & ? & ? & ? & \cdots & ? & X^{\delta m} &&& \\
& ? & ? & ? & ? & ? & ? & ? & \cdots & ? & X^{\delta m+1} && \\
&&&& \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\
&&& ? & ? & ? & ? & ? & ? & ? & \cdots & ? & X^{\delta m+t-1}
\end{array}\right)$$

Notice that the above matrix is lower triangular. This means that all its entries l_{ij} such that $j > i$ are zero. An element l_{ij} such that $j < i$ may or may not be zero. We have used the symbol "?" to denote a possibly nonzero entry whose value doesn't affect the calculation of the determinant.

The determinant of the above matrix can now be easily calculated by multiplying all the diagonal entries. We then have

$$det(\mathcal{L}) = N^{\frac{1}{2}\delta m(m+1)} X^{\frac{1}{2}\omega(\omega-1)}$$

We will now optimize the value t. This will also optimize the value $\omega = \delta m + t$, that is the dimension of the lattice basis. Remember our initial goal. We want to construct a lattice such that the following inequation is satisfied for a bound X that is as large as possible.

$$\|f(xX)\| \leq \frac{b^m}{\sqrt{\omega}}.$$

A sufficient condition for the above inequation is as we have already seen

$$\|f(xX)\| \leq 2^{\frac{\omega-1}{4}} det(\mathcal{L})^{\frac{1}{\omega}} \leq \frac{b^m}{\sqrt{\omega}} \Rightarrow det(\mathcal{L}) \leq \frac{b^{m\omega}}{\omega^{\frac{\omega}{2}}} 2^{-\frac{\omega(\omega-1)}{4}}.$$

The above inequality can be further simplified if we neglect the terms that only depend on ω. This is not a groundless simplification as $b \geq N^\beta$ and N is very large (in the order of 2^{1000}) whereas the lattice dimension ω is negligible compared to N (in the order of 100). So we can write the above condition as $det(\mathcal{L}) < b^{m\omega}$. Suppose now that we add at the end of the above matrix a new row vector corresponding to a polynomial $h_i(x)$.This increases the dimension of the matrix by 1 and the determinant by a factor of $X^{\omega'-1}$ (notice that the vectors corresponding to $h_i(x)$ only add X terms to the determinant and that the the diagonal element of the last row will by construction be $X^{\omega'-1}$ where ω' is the new dimension of the lattice.). Consider now the condition $det(\mathcal{L}) < b^{m\omega}$. It is not difficult to see that if the contribution to the determinant of the element added to the lattice is less than b^m then this condition will yield a bound X that is larger than the respective bound for a smaller dimension ω. This observation gives as a condition for the optimization of t. As long as the diagonal elements of the matrix corresponding to the polynomials $h_i(x)$, $i = 0, 1, ..., t-1$ do not grow larger than b^m, then we can benefit by increasing the bound X. Observe that $X^{\delta m} < X^{\delta m+1} < ... < X^{\delta m+t-1}$. This leads us to the following sufficient condition for the dimension of the lattice.

$$X^{\omega-1} < b^m.$$

Since by hypothesis $X^{\omega-1} < N^{(\frac{\beta^2}{\delta}-\epsilon)(\omega-1)}$ and $b \geq N^\beta$, this condition is satisfied for the choice

$$\omega \leq \frac{\delta}{\beta} m.$$

Thus the best value for ω would be $\omega = \frac{\delta}{\beta} m$.

According to 3.3 we can choose m as the maximum of$\left\{\frac{\beta^2}{\delta\epsilon}, \frac{7\beta}{\delta}\right\}$. This gives the following bound for the lattice dimension ω.

$$\omega = max\left\{\frac{\beta}{\epsilon}, 7\right\}.$$

Remark 3.2.8. Notice that the lattice dimension is polynomial in $\frac{1}{\epsilon}$. In addition, the bit-size of the entries in B can be bounded by

$$(\delta + m)logN \leq (\delta + \omega)logN$$

which means that LLL operates on basis B in time polynomial in $logN, \delta$ and $\frac{1}{\epsilon}$.

In the rest of the proof we show that LLL finds a vector which is sufficiently short and yields a sufficiently large bound X. In order to bound X by applying Lemma 3.2.6 we use the inequality

$$\|f(xX)\| \leq 2^{\frac{\omega-1}{4}} det(\mathcal{L})^{\frac{1}{\omega}} \leq \frac{b^m}{\sqrt{\omega}}.$$

Plugging in the value of the determinant and taking int consideration that $b \geq N^{\beta}$ the above inequality becomes

$$N^{\frac{\delta m(m+1)}{2\omega}} X^{\frac{\omega-1}{2}} \leq 2^{-\frac{\omega-1}{4}} \omega^{-\frac{1}{2}} N^{\beta m} \Rightarrow$$
$$X \leq 2^{-\frac{1}{2}} \omega^{-\frac{1}{\omega-1}} N^{\frac{2\beta m}{\omega-1} - \frac{\delta m(m+1)}{\omega(\omega-1)}}.$$

Now notice that for $\omega \geq 7$,$\omega^{-\frac{1}{\omega-1}} = 2^{-\frac{log\omega}{\omega-1}} \geq 2^{-\frac{1}{2}}$. Therefore the above condition simplifies to

$$X \leq \frac{1}{2} N^{\frac{2\beta m}{\omega-1} - \frac{\delta m(m+1)}{\omega(\omega-1)}}.$$

We have chosen $X = \frac{1}{2} N^{\frac{\beta^2}{\delta} - \epsilon}$. Hence it only remains to show that

$$\frac{\beta^2}{\delta} - \epsilon \leq \frac{2\beta m}{\omega - 1} - \frac{\delta m(m+1)}{\omega(\omega-1)}.$$

But we know that

$$\frac{2\beta m}{\omega-1} - \frac{\delta m(m+1)}{\omega(\omega-1)} \geq \frac{\omega-1}{\omega}\left(\frac{2\beta m}{\omega-1} - \frac{\delta m(m+1)}{\omega(\omega-1)}\right)$$
$$= \frac{2\beta m}{\omega} - \frac{\delta m^2(1+\frac{1}{m})}{\omega^2} = 2\frac{\beta^2}{\delta} - \frac{\beta^2}{\delta}(1+\frac{1}{m}) \geq \frac{\beta^2}{\delta} - \epsilon$$

where the in the last equality we have used that $\omega = \frac{\delta}{\beta} m$ and the last inequality holds because of the choice of m , $m \geq \frac{\beta^2}{\delta\epsilon}$ we have made in 3.3. He have proved the bound. The fact that we can find all solutions x_0 such that $|x_0| \leq X$ in time polynomial in $(\log N, \delta, \frac{1}{\epsilon})$ is an immediate result of remark 3.2.8 and the polynomial running time of LLL. □

Below we give two results that stem immediately from the previous theorem. We choose to state them here in order to highlight their importance. The second result is the initial result presented by Coppersmith [10] and is therefore stated as a theorem.

Corollary 3.2.9. Let N be an integer of unknown factorization, which has a divisor $b \geq N^{\beta}$. Furthermore, let $f(x)$ be a univariate, monic polynomial of degree δ and c_N be a function that is upper-bounded by a polynomial in $\log N$. Then we can find all solutions x_0 for the equation $f(x) \equiv 0 (mod\, b)$ with

$$|x_0| \leq c_N N^{\frac{\beta^2}{\delta}}$$

in time polynomial in $(\log N, \delta)$.

Proof. We apply theorem 3.2.7 with parameter choice $\epsilon = \frac{1}{\log N}$.The bound X for the absolute value of the solutions will then be:

$$\frac{1}{2} N^{\frac{\beta^2}{\delta} - \frac{1}{\log N}} = \frac{1}{2} N^{\frac{\beta^2}{\delta}} N^{-\frac{1}{\log N}} = \frac{1}{2} N^{\frac{\beta^2}{\delta}} \frac{1}{N^{\frac{1}{\log N}}} = \frac{1}{4} N^{\frac{\beta^2}{\delta}}$$

since $\frac{1}{N^{\frac{1}{\log N}}} = \frac{1}{2}$. In addition, theorem 3.2.7 says that we can find all the solutions x_0 such that $|x_0| < X$ in time polynomial in $(\log N, \delta, \frac{1}{\epsilon})$. For $\epsilon = \frac{1}{\log N}$ this means that we can find all the solutions x_0 in time polynomial in $(\log N, \delta)$. In order to find all roots that are the size at most $c_N N^{\frac{\beta^2}{\delta}}$ in absolute value, we divide the interval $[-c_N N^{\frac{\beta^2}{\delta}}, c_N N^{\frac{\beta^2}{\delta}}]$ into $4c_N$ subintervals of size $\frac{1}{2} N^{\frac{\beta^2}{\delta}}$ centered at some x_i. Then in order to find all the roots in the above interval, we just have to apply the method described in theorem 3.2.7 to the polynomials $f(x - x_i)$ and output the roots in each subinterval. □

Theorem 3.2.10 (**Coppersmith Theorem for Univariate Modular Equations**)
Let N be an integer of unknown factorization. Furthermore, let $f(x)$ be a univariate, monic polynomial of degree δ. Then we can find all solutions x_0 for the equation $f(x) \equiv 0 (mod\, N)$ with

$$|x_0| \leq N^{\frac{1}{\delta}}$$

in time polynomial in $(\log N, \delta)$.

Proof. Immediate application of corollary 3.2.9 where $c_N = 1$ and $b = N$. □

Remark 3.2.11. A question that arises naturally is: Can we improve the asymptotic bound $X = N^{\frac{1}{d}}$?In [12] Coppersmith tries to give an answer. Let $N = q^3$ where q is a prime and consider $p(x) = x^3 + Dqx^2 + Eq^2x$ with $D, E \in \mathbb{Z}$.If x_0 is any multiple of q, then clearly $p(x_0) \equiv 0 (mod\, N)$. Suppose now that we can achieve a bound $X = N^{\frac{1}{3}+\epsilon}$. Then the number of "small" roots is given by the inequation

$$|x_0| < X \Rightarrow |kq| < N^{\frac{1}{3}} N^{\epsilon} = qN^{\epsilon} \Rightarrow |k| < N^{\epsilon}$$

which gives exponentially many solutions $(2N^{\epsilon})$. We cannot hope to find these solutions using the above lattice techniques since we would need lattices of exponential dimension in order to construct polynomials h that would have exponentially many solutions (remember that h has at least as many roots over $\mathbb{Z}$ as f modulo N and that the number of integer roots of h is bounded by its degree which is equal to the dimension of the lattice that produces h). This observation gives cause for pessimism to our attempt to further improve the bound X using lattice techniques.

3.2.2 Extension to More than One Variables

A very natural question is whether one can extend Coppersmith's technique to the case of multivariate modular polynomial equations. More formally let $f(\vec{x}) = f(x_1, x_2, ..., x_k) \in \mathbb{Z}[x_1, ..., x_k]$ be a multivariate polynomial in k variables with integer coefficients. We are interested in finding solutions $\vec{y} = (y_1, ..., y_k)$ to the following modular equation

$$f(\vec{x}) = f(x_1, x_2, ..., x_k) = \sum_{i_1,...,i_k} a_{i_1,...,a_k} x_1^{i_1} ... x_k^{i_k} \equiv 0 \, (mod \, N). \tag{3.4}$$

In principle there is no problem in applying Coppersmith's technique from the previous section. That is, we can construct from $f(x_1, x_2, ..., x_k)$ a polynomial $h(x_1, x_2, ..., x_k)$ with the same "small" roots over the integers and not just modulo N. The following lemma due to Howgrave-Graham,is a direct generalization of lemma 3.2.2 to the multivariate case and states explicitly the conditions for the upper bounds $X_1, X_2, ..., X_k$ under which we can achieve the transformation of the modular equation to an equation over the integers.

Lemma 3.2.12 (**Howgrave-Graham for Multivariate Integer Polynomials**). Let $f(x_1, ..., x_k) \in \mathbb{Z}[x_1, ..., x_k]$ be a polynomial in k variables with at most ω monomials and let m be a positive integer. Suppose in addition that:

1. $f(x_1, ..., x_k) \equiv 0 (mod \, N^m)$ where $|x_i| < X_i, \quad i = 1, ..., k$
2. $\|f(x_1X_1, ..., x_kX_k)\| \leq \frac{N^m}{\sqrt{\omega}}$.

Then $f(x_1, ..., x_k) = 0$ holds over the integers.

Proof. The proof is completely analogous to the proof of lemma 3.2.2 and is therefore omitted. □

Again here the goal is to find the maximum bounds $X_1, X_2, ..., X_k$ so that all solutions $\vec{x} = (x_1, ..., x_k)$ such that $f(x_1, ..., x_k) \equiv 0(mod \, N^m)$ with $|x_i| < X_i$ can be efficiently found.

Everything seems to work in a completely similar way to the univariate case. The difference here lies to the fact that even if we manage to construct a polynomial $h(x_1, ..., x_k)$ with a small root over the integers, we still have to extract the integer roots of $h(x_1, ..., x_k)$.In contrast to the univariate case, there cannot be a polynomial algorithm that solves the above problem for the general multivariate case. To see that, consider the following example

$$f(x,y) = x + ay, \quad a \in \mathbb{Z}.$$

This polynomial has infinitely many integer solutions and thus we cannot recover them in polynomial time. In contrast, if f is univariate, the fundamental theorem of algebra says that the number of its integer solutions is bounded by its degree.

The key idea in order to overpass this problem is the following:
Take the first k coefficient vectors returned by LLL Algorithm that correspond to k polynomials instead of considering only the first vector. Then we can solve the system of k polynomials and compute their common roots.

Unfortunately,as we will later explain , this approach does not always lead to the recovery of the common roots.

Below,we describe in brief the method for the multivariate case and state the conditions under which we can recover the small roots for the equation $f(x_1,...,x_k) \equiv 0(mod\, N^m)$.Our presentation follows the presentation in [22].

Constructing the Lattice and Obtaining the Conditions

Let m and d be positive integers. We define the polynomials:

$$f_{r_1,...,r_k,j}(\vec{x}) = f_{r_1,...,r_k,j}(x_1,...,x_k) \in \mathbb{Z}[x_1,...,x_k]$$

by

$$f_{r_1,...,r_k,j}(\vec{x}) = N^{m-j} x_1^{r_1} \cdots x_k^{r_k} (f(\vec{x}))^j \tag{3.5}$$

where $0 \leq j \leq m$ and $r_i \geq 0$ for $i = 1,...,k$ are integers.Notice that by construction,if $\vec{y}$ is a solution of $f(\vec{x}) \equiv 0(mod\, N)$ then $\vec{y}$ is a root of $f_{r_1,...,r_k,j}(\vec{x})$ modulo N^m for all valid j and r_i. Moreover, for any fixed j, the polynomials of the form 3.5 with different $(r_1,...,r_k)$ values are linearly independent. So we can construct a lattice $\mathcal{L}$ of dimension ω and basis matrix B whose row vectors will be the coefficient vectors of the above polynomials. Thus each row of $\mathcal{L}$ will be of the form

$$f_{r_1,...,r_k,j}(x_1X_1,...,x_kX_k).$$

If we choose the values $(r_1,...,r_k,j)$ in a "convenient" way then we can construct a low triangular basis matrix B which greatly simplifies the computation of the lattice determinant. Which choice is convenient depends on the particular structure of the polynomial $f(x_1,...,x_k)$.

After constructing the lattice $\mathcal{L}$, we run the LLL Algorithm with input the basis B and consider k linearly independent vectors in $\mathcal{L}$ returned by LLL. These vectors correspond to k linearly independent polynomials $p_i(\vec{x})$ and by theorem 3.1.1 they satisfy

$$\|p_i(x_1X_1,...,x_kX_k)\| \leq c(i,\omega) det(\mathcal{L})^{\frac{1}{\omega-i+1}}, \quad for\, i = 1,...,k$$

where $c(i, \omega)$ is a function that depends only on i and ω. We now need a condition in order to detemine the bounds X_i, $i = 1, 2, ..., k$. A sufficient condition in order to be able to apply lemma 3.2.12 on each of the above polynomials is the following

$$\|p_k(x_1X_1, ..., x_kX_k)\| \leq c(k, \omega)det(\mathcal{L})^{\frac{1}{\omega-k+1}} \leq \frac{N^m}{\sqrt{\omega}}. \tag{3.6}$$

Notice here that the fact that $\|p_i(x_1X_1, ..., x_kX_k)\| \leq c(i, \omega)det(\mathcal{L})^{\frac{1}{\omega-i+1}}$ along with the fact that the terms $c(i, \omega)$ are negligible compared to the term $N^{\frac{1}{\omega-i+1}}$ implies that $\|p_i(x_1X_1, ..., x_kX_k)\| \leq c(k, \omega)det(\mathcal{L})^{\frac{1}{\omega-k+1}}$ is true whenever $\|p_k(x_1X_1, ..., x_kX_k)\| \leq c(k, \omega)det(\mathcal{L})^{\frac{1}{\omega-k+1}}$ and hence condition 3.6 is indeed sufficient. The bounds X_i can be calculated more easily if we ignore the terms $\sqrt{\omega}$ and $c(k, \omega)$ (which are negligible compared to $det(\mathcal{L})^{\frac{1}{\omega-k+1}}$ and N^m) in condition 3.6.

Recovering the Roots

After determining the bounds X_i via condition 3.6, we end up with k polynomials p_i, $i = 1, ..., k$ some of the roots of which are equal to the "small" (by small we mean the roots such that $|x_i| < X_i$ for the bounds X_i derived from 3.6) roots of $f(x_1, ..., x_k) \equiv 0 (mod\, N^m)$.

In order to recover the tuples $(x_1, x_2, ..., x_k)$ which correspond to small solutions ($|x_i| < X_i$) of the initial multivariate modular equation, we have to solve the system of k non-linear equations in k variables.So far this method seems to work according to the univariate case. The major (theoretical) drawback of this approach is that there is no known method to solve the above non-linear system.

We noted before that the polynomials that correspond to the coefficient vectors returned by LLL are linearly independent. Unfortunately this condition is not sufficient to guarantee that we can recover the common roots $(x_1, ..., x_k)$ of the system of k equations, since the equations are non linear. Here, apart from linear indpendence, we need algebraic independence. This means that there should be no pair of polynomials (p_i, p_j) such that $p_i(x_1, ..., x_k) = s(x_1, ..., x_k)p_j(x_1, ..., x_k)$ for an integer polynomial $s(x_1, ..., x_k)$. In cases where the polynomials are algebraically independent we can recover the common solutions $(x_1, x_2, ..., x_k)$ using resultant computations. Below we describe the procedure in brief.

Let $p_1, ..., p_k$ be k polynomials in the variables $x_1, ..., x_k$. We first compute the $k - 1$ resultants

$$g_1 = res_{x_1}(p_1, p_2), g_2 = res_{x_1}(p_2, p_3), \ldots, g_{k-1} = res_{x_1}(p_{k-1}, p_k)$$

which are $(k - 1)$-variate polynomials in the variables $x_2, ..., x_k$. The elimination of variable x_1 leads to a system of $k - 1$ non-linear equations in $k - 1$ variables. If none of the resultants $g_1, g_2, ..., g_{k-1}$ is the zero polynomial, we can keep on by eliminating x_2:

$$h_1 = res_{x_2}(g_1, g_2), h_2 = res_{x_2}(g_2, g_3), ..., h_{k-2} = res_{x_2}(g_{k-2}, g_{k-1}).$$

We can proceed in this way as long as all the resultants in each step are nonzero polynomials. At the last step we will have eliminated all but the last variable x_k. This means that the last resultant is a univariate polynomial in x_k and can be therefore be solved using standard root finding algorithms.

We summarize Coppersmith's method for multivariate modular polynomial equations:

- We first construct k different k-variate polynomials $p_1, p_2, ..., p_k$ with some common small roots.This construction as described above, is analogous to the univariate case.

- We then try to extract the common roots using resultant computations.

- If a resultant produced during the above step equals the zero polynomial, then the procedure fails.Otherwise we find the roots x_k of the last resultant and by backsolving for all possible roots obtained so far, we finally get all possible k-tuples $(x_1, ..., x_k)$ that are the roots of the initial system of k non-linear equations. We then have to check each of these tuples and accept only those that satsify $f(x_1, ..., x_k) \equiv 0(mod N^m)$.

Remark 3.2.13. The negative aspects of Cppersmith's method for solving multivariate modular equations should by no means be overemphasized. Although the above method is heuristic rather than provable, it has found a large number of applications in Crytpography.The experiments carried out so far by various researchers show that the resultant computations are in many situations a very useful method in order to extract roots of multivariate polynomials over the integers. We will demonstrate the use of the above method in a concrete cryptanalytic application later in the thesis.

Remark 3.2.14. This method will be frequently refered to as the *resultant heuristic*.The fact that the resultant heuristic is very useful in practice enables us to frequently make the assumption that the heuristic always works and therefore state the consequent results as theorems. However this assumption, when made, will be explicitly stated as such throughout this thesis.

In light of this remark, it would be very interesting if we could find explicit conditions under which Coppersmith's method for the multivariate case succeeds to find small roots. Till now, no such conditions are known and therefore the problem of finding a provable method that leads to explicit conditions remains open.

3.3 Integer Polynomial Equations

In the previous section we presented a method for finding small roots to modular equations.More specifically,we were given a polynomial $f(\vec{x}) = f(x_1, x_2, ..., x_k)$

where $k \in \mathbb{Z}$ such that $k \geq 1$ and a modulus N and we were interested in finding small root-vectors $\vec{x_0} \in \mathbb{Z}^k$ such that $f(\vec{x_0}) \equiv 0(mod\, N)$.

In this section we are interested in small roots over the integers and not modulo N. We are given a polynomial $f(\vec{x}) = f(x_1, ..., x_k)$ with integer coefficients and we are searching for small root-vectors, that is vectors $\vec{x_0} \in \mathbb{Z}^k$ such that $f(\vec{x_0}) = 0$ .Using a standard root finding algorithm, we can find all integers roots (and not only the small ones) in the univariate case. However no such algorithm is known for the case where $k \geq 2$. Instead, we can find small integer roots of multivariate polynomials using lattice reduction techniques similar to the ones presented in the previous section. Here we present in detail a method for finding small roots to bivariate integer equations. The bivariate case is of great importance since many problems in cryptography can be reduced to the problem of finding a small solution to a bivariate integer equation.At the end of this subsection we sketch in brief the extension of the bivariate method to the k-variate case where $k \geq 3$.

3.3.1 Bivariate Case

In 1996, Coppersmith [9, 11] proposed a method for finding small roots to bivariate integer polynomials. This method was based on lattice reduction techniques too, but his approach (as in the univariate case) was difficult to understand. In 2004, Coron [14] presented a simpler approach to Coppersmith's method. This simplification is analogous to the simplification brought by Howgrave-Graham to Coppersmith's method for finding small roots to univariate modular equations.Here we will state both Coppersmith's and Coron's result (which is slightly weaker) and present the proof according to Coron's approach.

Let us first introduce the problem a little more formally. Consider the following polynomial in two variables with integer coefficients:

$$p(x, y) = \sum_{i,j} p_{i,j} \cdot x^i y^j.$$

We are interested in finding all the integer pairs (x_0, y_0) such that $p(x_0, y_0) = 0$. In general there is no efficient algorithm that finds such pairs. However, Coppersmith [9] show that one can efficiently find small root pairs of the equation $p(x, y) = 0$. More specifically Coppersmith proves the following theorem.

Theorem 3.3.1 (Coppersmith's Theorem for Bivariate Integer Equations) *Let $p(x, y)$ be an irreducible* [1] *polynomial in two variables over $\mathbb{Z}$, of maximum degree δ in each variable separately. Let X, Y be upper bounds on the desired integer*

[1]In general, a polynomial is said to be *irreducible* if it cannot be factorized into the product of two polynomials of lesser degree.

For bivariate polynomials, a bivariate polynomial $f(x, y)$ with integer coefficients is irreducible if there are no integer (bivariate) polynomials $f_1(x, y), f_2(x, y)$ with $f_1(x, y) \neq 1$ and $f_2(x, y) \neq 1$ such that $f(x, y) := f_1(x, y) \cdot f_2(x, y)$.

solution (x_0, y_0).Let W be the absolute value of the largest entry in the coefficient vector of p(xX,yY),that is $W = \max_{i,j}|p_{i,j}|X^iY^j$. If $XY \leq W^{\frac{2}{3\delta}}$, then in time polynomial in log W and 2^δ we can find all integer pairs (x_0, y_0) such that $p(x_0, y_0) = 0, |x_0| \leq X$ and $|y_0| \leq Y$.

Coppersmith's proof for the above case is quite difficult to understand. What makes his analysis harder, is the fact that the lattices used throughout the analysis are not full rank , fact that greatly complicates the calculation of the determinant and the derivation of the upper bounds on the roots.

As we mentioned, Coron [14] presented a simpler approach for finding small roots to bivariate integer polynomials. Coron uses full rank lattices that admit a triangular basis. His approach enjoys two major advantages:

- The conditions that yield the bounds X, Y on the solutions can be easily derived independently of the specific shape of the polynomial p and
- It is straightforward to heuristically extend this approach to more than two variables.

However, Coron's approach brings slightly weaker results than Coppersmith's, since polynomial time execution is guaranteed only for the case $XY \leq W^{\frac{2}{3\delta}-\epsilon}$ for a fixed $\epsilon > 0$ which is apparently a weaker condition. We first illustrate Coron's method with an example (also found in [14]).

Method illustration

Consider the polynomial

$$p(x, y) = a + bx + cy + dxy, \quad a, d \neq 0.$$

Assume that $p(x, y)$ is irreducible and has a small root (x_0, y_0) which we want to recover.We are interested in pairs (x_0, y_0) such that $|x_0| < X$ and $|y_0| < Y$ for some bounds X, Y that we will soon determine. W is defined as $W = max\{|a|, |b|X, |c|Y, |d|XY\}$. Finally, as usual,for the polynomial $h(x, y) = \sum_{i,j} h_{ij}x^iy^j$,we define

$$\|h(x, y)\|^2 := \sum_{i,j} |h_{ij}|^2 \quad and\, \|h(x, y)\|_\infty := \max_{i,j} |h_{ij}|.$$

This implies that an equivalent definition for W is

$$W = \|p(xX, yY)\|_\infty. \tag{3.7}$$

We now fix a number n such that:

$$W \leq n < 2 \cdot W \quad and \quad gcd(n, a) = 1. \tag{3.8}$$

Consider also the polynomials:

$$q_{00}(x,y) = a^{-1}p(x,y)\, mod\, n = 1 + b'x + c'y + d'xy$$
$$q_{10}(x,y) = nx,\ q_{01}(x,y) = ny \ and\ q_{11}(x,y) = nxy.$$

It is important to notice that $q_{ij}(x_0,y_0) = 0\,(mod\, n)$ for each of the above q_{ij}.This means that each linear integer combination of the above polynomials yields a polynomial h that is a root container of p modulo n. Consider the lattice $\mathcal{L}$ generated by all linear integer combination of the coefficient vectors of the above polynomials. A basis B for the lattice is obviously the following (here again the coefficient vectors form the rows of B).

$$B = \begin{bmatrix} 1 & b'X & c'Y & d'XY \\ & nX & & \\ & & nY & \\ & & & nXY \end{bmatrix}.$$

We would like to find a short linear combination of the above polynomials in order to be able to apply lemma 3.2.12 for the special case where $k = 2$.Applying LLL Algorithm with input basis B, theorem 3.1.1 guarantees an ouput B' such that $\|b_1\| \leq 2^{\frac{3}{4}} det(B)^{\frac{1}{4}}$. Here, it is trivial to see that $det(B) = n^3X^2Y^2$. Thus LLL returns a polynomial h such that

$$\|h(xX,yY)\| \leq 2^{\frac{3}{4}} n^{\frac{3}{4}} (XY)^{\frac{1}{2}} < 2n^{\frac{3}{4}}(XY)^{\frac{1}{2}} \quad \text{and} \ h(x_0,y_0) = 0\,(mod\, n). \tag{3.9}$$

Suppose now that $XY < \frac{n^{\frac{1}{2}}}{16}$. This along with the above inequality means that

$$\|h(xX,yY)\| < \frac{n}{2} < W = \|p(xX,yY)\|_\infty \leq \|p(xX,yY)\|. \tag{3.10}$$

Notice first that by the construction of the lattice $\mathcal{L}$, h is of the same shape as p which means that all its monomials have one of the forms c, cx, cy or cxy where c is an integer coefficient (not the same for each monomial).In addition p was assumed irreducible.This means that h is a multiple of p only if there exist an integer t such that $h(x,y) = tp(x,y)$. But inequality 3.10 overrules this possibility too since it would imply that $\|h(xX,yY)\| = |t|\|p(xX,yY)\| \geq \|p(xX,yY)\|$ which yields a contradiction.Finally notice that the conditions $\|h(xX,yY)\| < \frac{n}{2}$ and $h(x_0,y_0) = 0\,(mod\, n)$ imply that $h(x_0,y_0) = 0$ over the integers.

Let us now summarize the above observations. We have ended up with two polynomials h, p that are provably algebraically independent and that have the same small integer roots. We can therefore compute the nonzero resultant

$$Q(x) = res_y(h(x,y), p(x,y))$$

and recover a root x_0 such that $Q(x_0) = 0$ using any standard root finding algorithm. A root y_0 can then be recovered by solving $p(x_0,y) = 0$. If $XY < \frac{W^{\frac{1}{2}}}{16}$ then $XY < \frac{n^{\frac{1}{2}}}{16}$ since $W \leq n$. Thus we have showed that if $XY < \frac{W^{\frac{1}{2}}}{16}$ then one can find in polynomial time in $\log W$ all integer pairs (x_0, y_0) such that $p(x_0, y_0) = 0, |x_0| \leq X$ and $|y_0| \leq Y$.

Of course , the bound is weaker than the bound given in theorem 3.3.1 and it has been proved only for a specific shape for p. Coron [14] has improved the bound to the desired value by adding more multiples of $p(x, y)$ and increasing the dimension of the lattice.

Proving the Main Theorem

For his proof Coron first proves some useful lemmas.

Lemma 3.3.2. Let $a(x, y), b(x, y)$ be two nonzero polynomials over $\mathbb{Z}$ of maximum degree d separately in x and y, such that $b(x, y)$ is a multiple of $a(x, y)$ in $\mathbb{Z}[x, y]$. Then

$$\|b\| \geq 2^{-(d+1)^2} \cdot \|a\|_\infty$$

Proof. Mignotte [31] proved that if $f(x)$ and $g(x)$ are two nonzero polynomials over the integers such that $\deg f \leq k$ and f divides g in $\mathbb{Z}[x]$ then

$$\|g\| \geq 2^{-k} \cdot \|f\|_\infty.$$

Let $f(x) = a(x, x^{d+1})$. Then it is not difficult to see that $\deg f \leq (d+1)^2$ and that $f(x), a(x, y)$ have the same set of nonzero coefficient which implies that $\|f\|_\infty = \|a\|_\infty$.Similarly if we define $g(x) = b(x, x^{d+1})$,then $\|g\| = \|b\|$. In addition by construction of $f(x), g(x)$ and by hypothesis, we get that $f(x)$ divides $g(x)$ in $\mathbb{Z}[x]$.If we now apply Mignotte's result the we prove the desired inequality. □

Lemma 3.3.3. Let $a(x, y)$ and $b(x, y)$ satisfy the same conditions as in lemma 3.3.2. Assume in addition that $a(0, 0) \neq 0$ and $b(x, y)$ is divisible by a nonzero integer r such that $\gcd(r, a(0, 0)) = 1$. Then $b(x, y)$is divisible by $r \cdot a(x, y)$ and

$$\|b\| \geq 2^{-(d+1)^2} \cdot |r| \cdot \|a\|_\infty.$$

Proof. Since $b(x, y)$ is a multiple of $a(x, y)$, there exists a polynomial $q(x, y)$ such that $b(x, y) = q(x, y)a(x, y)$.We will show that r divides $q(x, y)$. Suppose on the contrary that r does not divide $q(x, y)$ and consider the smallest (lexicographically) pair (i, j) such that the coefficient q_{ij} of the term $x^i y^j$ is not divisible by r. Then we have that $b_{ij} = q_{ij} \cdot a(0, 0) \, mod \, r$ where b_{ij} is the coefficient of $x^i y^j$ (notice that the other terms that contribute to the coefficient b_{ij} are of the form $q_{kl} a_{(i-k)(j-l)}$ where the pair (k, l) is assumed smaller than (i, j) in lexicographic order and therefore r/q_{kl}).Since $a(0, 0)$ is assumed invertible modulo r and r divides by hypothesis every b_{ij} this implies that $q_{ij} \equiv 0 \pmod r$ which yields a contradiction.This means that r divides $q(x, y)$ and thus $r \cdot a(x, y)$ divides $b(x, y)$. Finally the inequality is a direct application of lemma 3.3.2 and the fact that $\|r \cdot a\|_\infty = |r| \cdot \|a\|_\infty$. □

Coron proves the following (slightly weaker) theorem

Theorem 3.3.4
Let $p(x,y)$ be an irreducible polynomial in two variables over $\mathbb{Z}$, of maximum degree δ in each variable separately. Let X,Y be upper bounds on the desired integer solution (x_0,y_0).Let $W = \max_{i,j} |p_{i,j}|X^iY^j$. If for some $\epsilon > 0$

$$XY \leq W^{\frac{2}{3\delta}-\epsilon}, \tag{3.11}$$

then in time polynomial in $(\log W, 2^\delta)$ one can find all integer pairs (x_0,y_0) such that $p(x_0,y_0) = 0, |x_0| \leq X$ and $|y_0| \leq Y$.

Proof. We consider the case where $p_{00} \neq 0$ and $\gcd(p_{00}, XY) = 1$. Coron in [14, Appendix A] provides an approach that obviates the need of these assumptions.

We will again try to convert the integer equation to a bivariate modular equation and then obtain a new bivariate polynomial h that is algebraically independent to p and has the same small roots over the integers. Using resultant computations, we will then recover the common roots.For this,select an integer $k \geq 0$ and let $\omega = (\delta + k + 1)^2$, ω will be the dimension of the lattice constructed. Generate an integer u such that $\sqrt{\omega}\cdot 2^{-\omega}\cdot W \leq u < 2W$ and $\gcd(p_{00}, u) = 1$. Let $n = u\cdot(XY)^k$. This implies that $\gcd(p_{00}, n) = 1$ and

$$\sqrt{\omega}\cdot 2^{-\omega}\cdot (XY)^k \cdot W \leq n < 2\cdot (XY)^k\cdot W. \tag{3.12}$$

We define the polynomials $q(x,y)$ as follows:

$$q(x,y) = p_{00}^{-1}\cdot p(x,y) \; mod\, n = 1 + \sum_{(i,j)\neq(0,0)} a_{ij}x^iy^j.$$

Define now the polynomials

$$q_{ij}(x,y) = \begin{cases} x^iy^jX^{k-i}Y^{k-j}q(x,y) & \text{if } 0 \leq i,j \leq k \\ x^iy^jn & \text{if } (i,j) \in [0,\delta+k]^2 \backslash [0,k]^2 \end{cases}$$

Notice that for all $(i,j) \in [0,\delta+k]^2$ we have that $q_{ij}(x_0,y_0) = 0\, mod\, n$ and $(XY)^k/q_{ij}(xX,yY)$. This means that if $h(x,y)$ is a linear integer combination of the polynomials $q_{ij}(x,y)$ then $h(x_0,y_0) = 0\, mod\, n$ and $(XY)^k/h(xX,yY)$. In addition , by construction, $h(x,y)$ has maximum degree $\delta + k$ independently in x and y and thus it is the sum of at most ω monomials. We will now search for a polynomial $h(x,y)$ such that the coefficients of $h(xX,yY)$ are small enough.This would enable us to convert the modular equation $h(x,y) = 0\, mod\, n$ to an equation over the integers by applying lemma 3.2.12. The condition we need in order to apply the above lemma is

$$\|h(xX,yY)\| < \frac{n}{\sqrt{\omega}}. \tag{3.13}$$

In addition small coefficients of $\|h(xX,yY)\|$ would possibly mean that $h(xX,yY)$ is not a multiple of $p(xX,yY)$ which in turn implies that $h(x,y)$ is not a multiple of $p(x, y)$ and thus their resultant is nonzero. Let us now derive the condition for the latter case. Suppose that $h(xX, yY)$ is a multiple of $p(xX, yY)$ and define $r = (XY)^k$. Since $(XY)^k/h(xX, yY)$, $\gcd((XY)^k, p_{00}) = 1$ and $p_{00} \neq 0$ we can apply lemma 3.3.3 where $a(x, y) = p(xX, yY)$ and $b(x, y) = h(xX, yY)$ and get

$$\|h(xX, yY)\| \geq 2^{-(\delta+k+1)^2} \cdot (XY)^k \cdot W$$

where, as usually, $W = \|p(xX, yY)\|_\infty$. Thus the inversion of the above inequality is sufficient to ensure that $h(xX, yY)$ is not a multiple of $p(xX, yY)$. Notice that $(\delta + k + 1)^2 = \omega$ which finally gives the second condition

$$\|h(xX, yY)\| < 2^{-\omega} \cdot (XY)^k \cdot W \tag{3.14}$$

We are searching for a polynomial $h(x, y)$ that satisfies both conditions. This requirement is reduced to the satsifaction of inequality 3.14 since this inequality ensures the satisfaction of the inequality 3.13 too (this is obvious by inequality 3.12).

In order to find the polynomial $h(x, y)$ we form the lattice $\mathcal{L}$ spanned by the polynomials $q_{ij}(xX, yY)$. Since $q_{ij}(x, y)$ have maximum degree $\delta + k$ separately in x, y, the polynomials obtained as linear combinations of $q_{ij}(xX, yY)$ have at most $(\delta + k + 1)^2 = \omega$ coefficients.Moreover there are $(\delta + k + 1)^2 = \omega$ in total polynomials $q_{ij}(x, y)$. This gives a full rank lattice of dimension ω. If we arrange these polynomials "conveniently", we can form a triangular basis matrix B of $\mathcal{L}$.Below we give such a formation for the parameter values $\delta = k = 1$.

	1	x	y	xy	x^2	x^2y	y^2	xy^2	x^2y^2
XYq	XY	$a_{10}X^2Y$	$a_{01}XY^2$	$a_{11}X^2Y^2$					
xYq		XY		$a_{01}XY^2$	$a_{10}X^2Y$	$a_{11}X^2Y^2$			
Xyq			XY	$a_{10}X^2Y$			$a_{01}XY^2$	$a_{11}X^2Y^2$	
xyq				XY		$a_{10}X^2Y$		$a_{01}XY^2$	$a_{11}X^2Y^2$
x^2n					X^2n				
x^2yn						X^2Yn			
y^2n							Y^2n		
xy^2n								XY^2n	
x^2y^2n									X^2Y^2n

A simple but tedious calculation shows that

$$det(\mathcal{L}) = (XY)^{\frac{(\delta+k)(\delta+k+1)^2+k(k+1)^2}{2}} n^{\delta(\delta+2k+2)}. \tag{3.15}$$

Using LLL and theorem 3.1.1 we can find in time polynomial in $(\log W, \omega)$ a nonzero polynomial $h(x, y)$ such that

$$\|h(xX, yY)\| \leq 2^{\frac{\omega-1}{4}} \cdot det(\mathcal{L})^{\frac{1}{\omega}}. \tag{3.16}$$

In order for a polynomial $h(x, y)$ to satisfy conditions 3.13 and 3.14 a sufficient condition is

$$2^{\frac{\omega-1}{4}} \cdot det(\mathcal{L})^{\frac{1}{\omega}} < 2^{-\omega} \cdot (XY)^k \cdot W \tag{3.17}$$

Plugging in the specific value of each term, we find that inequality 3.17 is satisfied when:

$$XY < 2^{-\beta} W^\alpha \tag{3.18}$$

where

$$\alpha = \frac{2(k+1)^2}{(\delta+k)(\delta+k+1)^2 - k(k+1)^2} \tag{3.19}$$

$$\beta = \frac{10}{4} \cdot \frac{(\delta+k+1)^4 + (\delta+k+1)^2}{(\delta+k)(\delta+k+1)^2 - k(k+1)^2}. \tag{3.20}$$

For $\delta \geq 1$ and $k \geq 0$ we have that

$$\alpha \geq \frac{2}{3\delta} - \frac{2}{3 \cdot (k+1)} \quad and \quad \beta \leq \frac{4k^2}{\delta} + 13 \cdot \delta.$$

If we take $k = \lfloor 1/\epsilon \rfloor$, the above inequalities along with 3.18 give the following condition for XY

$$XY < W^{\frac{2}{3\delta}-\epsilon} \cdot 2^{-\frac{4}{\delta \cdot \epsilon^2} - 13\delta}. \tag{3.21}$$

We have therefore managed to construct a polynomial $h(x,y)$ which possesses the two following properties:

(a) $h(x_0,y_0) = 0$ over the integers for all the pairs (x_0,y_0) such that $|x_0| < X, |y_0| < Y$ where $XY < W^{\frac{2}{3\delta}-\epsilon} \cdot 2^{-\frac{4}{\delta \cdot \epsilon^2} - 13\delta}$ and

(b) it is not a multiple of the irreducible polynomial $p(x,y)$.

This means that the resultant

$$Q(x) = res_y(h(x,y), p(x,y))$$

is a nonzero polynomial such that $Q(x_0) = 0$. We can therefore recover a root x_0 using any standard root finding algorithm. A root y_0 can then be recovered by solving $p(x_0,y) = 0$. It is important to note here that the above algorithm for finding small roots to bivariate integer polynomials runs in time polynomial in $(\log W, \delta, \frac{1}{\epsilon})$.

If we exhaustively search the high order $\frac{4}{\delta \cdot \epsilon^2} + 13\delta$ bits of x_0 and apply the above algorithm for each possible value, then we take the bounds given by 3.11.For a fixed ϵ the running time of the algorithm is polynomial in $(\log W, 2^\delta)$. □

As the theorem states, the condition 3.11 is sufficient for the case where $p(x,y)$ has maximum degree δ separately in x and y. Coron [14] also gives a theorem for the case where $p(x,y)$ has a total degree δ in x and y.

Theorem 3.3.5
Let $p(x,y)$ be an irreducible polynomial in two variables over $\mathbb{Z}$, of total degree δ in both variables x and y. Let X, Y be upper bounds on the desired integer solution (x_0,y_0).Let $W = \max_{i,j} |p_{i,j}| X^i Y^j$. If for some $\epsilon > 0$

$$XY \leq W^{\frac{1}{\delta}-\epsilon}, \tag{3.22}$$

then in time polynomial in $(\log W, 2^\delta)$ one can find all integer pairs (x_0,y_0) such that $p(x_0,y_0) = 0, |x_0| \leq X$ and $|y_0| \leq Y$.

Proof. The proof is analogous to the proof of theorem 3.3.4. More details can be found in [14, Appendix B]. □

3.3.2 Multivariate Case

The above method can be extended to integer polynomial equations with more than two variables much like the univariate modular equation method was extended to handle modular equations with more than one variables. Of course, the extension is heuristic only. In this subsection we will briefly outline the method.

Let $p(x_1, ..., x_l) \in \mathbb{Z}[x_1, ..., x_l]$ be a polynomial in l variables with degree δ in each variable. Our goal is to find all the integer solutions $\vec{y}$ of $p(x_1, ..., x_l) = 0$ such that $|y_i| < X_i$ for $i = 1, ..., l$.As in the bivariate case, we construct an integer n such that $(X_1 X_2 \cdots X_l)^k / n$ for some integer $k \geq 0$ and a polynomial $q(x_1, ..., x_l)$ such that $q(y_1, ..., y_l) = 0 \bmod n$ and $q(0, ..., 0) = 1$. We then consider the lattice $\mathcal{L}$ generated by all linear integer combinations of the polynomials

$$x_1^{r_1} \cdots x_l^{r_l} X_1^{k-r_1} \cdots X_l^{k-r_l} q(x_1 X_1, ..., x_l X_l) \text{ for } \quad 0 \leq r_1, ..., r_l \leq k$$

and the polynomials

$$(x_1 X_1)^{r_1} \cdots (x_l X_l)^{r_l} \cdot n \text{ for } \quad (r_1, ..., r_l) \in [0, \delta + k]^l \backslash [0, k]^l.$$

If the ranges $X_1, ..., X_l$ are small enough, then by using LLL, we can find polynomial $h_1(x_1, ..., x_l)$ such that $h_1(y_1, ..., y_l) = 0$ over $\mathbb{Z}$ and $h_1(x_1, ..., x_l)$ is not a multiple of $p(x_1, ..., x_l)$. However, this is not enough for the case where $l \geq 3$ as the 2 polynomials h_1, p are not enough to recover the pairs $(y_1, ..., y_l)$. We need at least l polynomials.Hence, apart from $h_1(x_1, ..., x_l)$ we also need to consider all the polynomials that correspond to the smallest $l - 1$ elements returned by LLL.For small enough bounds $X_1, ..., X_l$ we can obtain $l - 1$ polynomials $h_1(x_1, ..., x_l), ..., h_{l-1}(x_1, ..., x_l)$ which satisfy the following conditions:

1. $h_i(y_1, ..., y_l) = 0$ for $i = 1, 2, ..., l - 1$.The conditions for the bounds $X_1, ..., X_l$ are obtained by applying lemma 3.2.12 to all polynomials $h_i(x_1 X_1, ..., x_l X_l)$. But the norms of these polynomials can be bounded by theorem 3.1.1 (inequality 3.1). The combination of these two theorems gives sufficient conditions for the bounds $X_1, ..., X_l$.

2. $h_i(x_1, ..., x_l)$ is not a multiple of $p(x_1, ..., x_l)$ where $i = 1, 2, ..., l - 1$.The ranges $X_1, ..., X_l$ that satisfy this condition are dictated by a generalization of lemma 3.3.3.

After finding the polynomials $h_1(x_1, ..., x_l), ..., h_{l-1}(x_1, ..., x_l)$ we use resultant computations between the l polynomials $p(x_1, ..., x_l), h_1(x_1, ..., x_l), ..., h_{l-1}(x_1, ..., x_l)$ in order to obtain a polynomial $f(x)$ such that $f(y_1) = 0$. The l-tuples $(y_1, ..., y_l)$ are then recovered by backsolving. The major drawbach of this method is that the algebraic independence of each pair $h_i(x_1, ..., x_l), p(x_1, ..., x_l)$ for $i = 1, ..., l - 1$

does not mean that $h_i(x_1, ..., x_l)$, $h_j(x_1, ..., x_l)$ are algebraically independent too and thus the method does not guarantee that all the resultant computations will lead to nonzero polynomials. This makes the method heuristic only.

Chapter 4

A Positive Lattice Application to RSA

In 1976,Whitfield Diffie and Martin Hellman [17] introduced the idea of **Public-Key Cryprography.** In their paper, Diffie and Hellman proposed the use of different keys for encryption and decryption and introduced the notion of trapdoor one-way functions. A *trapdoor one-way function* is a function that can be computed efficiently but for which there is no efficient algorithm that inverts the function without the knowledge of a certain trapdoor. Diffie and Hellman only presented the properties such a function should possess and did not provide any specific example of such a function.

One year later, in 1977, Ronald Rivest ,Adi Shamir and Leonard Adleman in their famous paper "A method for Obtaining Digital Signatures and Public-Key Cryptosystems" [34] presented the well-known RSA Cryptosystem which consists the first implementation of a trapdoor one-way function in Public-Key Cryptography.Since then , RSA has become probably the most commonly used Cryptosystem in applications where providing privacy and ensuring authenticity of digital data are crucial. Some typical RSA applications include ensuring secure remote login sessions, privacy and authenticity of email and electronic credit-card payment systems robustness.

The rest of the thesis is exclusively devoted to the RSA Cryptosystem. In particular, we study applications that are related to lattice methods. We will use the knowledge acquired in previous chapters to illustrate lattice-based approaches to RSA-related problems. Throughout this and the next chapter, we will frequently invoke results presented in the previous chapters and especially those which concern solving modular or integer polynomial equations. It is therefore important for the reader to fully understand the preceding material before proceeding to this chapter.

We begin this chapter by presenting an introduction to Cryptosystems and a formal definition of RSA (section 4.1). In section 4.2 we describe a recently discovered positive application of lattices to RSA. More specifically, we present a lattice-based method that establishes the deterministic polynomial time equivalence between computing the RSA secret exponent d and factoring RSA modulus N.

4.1 The RSA Cryptosystem

Before presenting RSA , we first give a formal definition of the term *Cryptosystem* as defined in [37]. The following definition applies both to private-key (symmetric) and to public-key (assymetric) Cryptosystems.

Definition 4.1.1 (Cryptosystem)
A **cryptosystem** is a five-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, where the following conditions are satisfied:

1. $\mathcal{P}$ is a finite set of possible *plaintexts.*
2. $\mathcal{C}$ is a finite set of possible *ciphertexts.*
3. $\mathcal{K}$, the *keyspace*, is a finite set of possible *keys.*
4. For each $K \in \mathcal{K}$, there is an *encryption rule* $e_K \in \mathcal{E}$, and a corresponding *decryption rule* $d_K \in \mathcal{D}$.Each $e_K : \mathcal{P} \rightarrow \mathcal{C}$ and $d_K : \mathcal{C} \rightarrow \mathcal{P}$ are functions such that $d_K(e_K(x)) = x$ for every plaintext element $x \in \mathcal{P}$.

In symmetric Cryptosytems, the key for encryption and decryption is the same.In contrast,in public-key (assymetric) encryption systems, each entity A (usually referred to as Alice in bibliography) has a public key e and a corresponding private key d.In secure cryptosystems, the task of computing d given e is computationally infeasible. The public key defines an encryption transformation E_e, while the private key defines the associated decryption transformation D_d. An entity B (usually referred to as Bob), wishing to send a message m to A obtains an authentic copy of A's public key e, uses the encryption transformation to produce a ciphertext $c = E_e(m)$ and transmits c to A. To decrypt c, A applies the decryption transformation to obtain the original message $m = D_d(c)$.

The main objective of public-key encryption is to provide privacy and confidentiality.The public key e need not be kept secret whereas the private key d is known only to the legitimate entity. The main advantage of public key Cryptosystems over symmetric Cryptosystems is that providing authentic public keys is generally easier than distributing secret keys securely. However, Public-Key cryptosystems are typically substantially slower than the symmetric ones.That's why public-key encryption is most commonly used in practice for the transmission of keys subsequently used for bulk data encryption by symmetric algorithms.

Below we describe the RSA Cryptosystem, the most widely used public-key Cryptosystem. In algorithm 5 we present the generation of the parameters (keys) of RSA Cryptosystem while in algorithms 6 and 7, we present the encryption and decryption process respectively.

```
Algorithm 5: RSA-Key Generation
  Input: The bitsize of the modulus $N$.
  Output: A public key $(N, e)$ and a private key $d$.
  begin
      Step 1.Generate two large random and distinct primes $p$ and $q$ of
      about the same bitsize.
      Step 2.Compute $N = p \cdot q$ and $\phi(N) = (p-1) \cdot (q-1)$.
      Step 3.Select a random integer $e, 1 < e < \phi(N)$ such that
      $\gcd(e, \phi(N)) = 1$.
      Step 4.Use the extended Euclidean algorithm to compute the
      unique integer $d, 1 < d < \phi(N)$, such that $e \cdot d \equiv 1 \ (mod\, \phi(N))$.
      Step 5. A's public key is $(N, e)$; His private key is $d$.
  end
```

The integers e and d in RSA Key Generation are called the *encryption exponent* and the *decryption exponent* respectively while N is called the *modulus.*

Remark 4.1.2. In the above algorithm we have restricted the values of e, d to the interval $[1, \phi(N)]$. We just mention that this is the typical values for the keys e, d produced by the key generation process. However, each entity A can choose $e, d > \phi(N)$ and the encryption and decryption processes work as well provided that $e \cdot d \equiv 1 \ (mod\, \phi(N))$.

```
Algorithm 6: RSA Encryption
  Input: Public Key $(N, e)$ and plaintext $m$.
  Output: Ciphertext $c$ corresponding to plaintext $m$.
  begin
      B (the sender) should do the following:
      Step 1.Obtain A's authentic public key $(N, e)$.
      Step 2.Represent the message he wants to send as an integer $m$ in
      the interval $[0, N-1]$.
      Step 3. Compute $c = m^e \ (mod\, N)$.
      Step 4.Send the ciphertext $c$ to A.
  end
```

Remark 4.1.3. This is the initial definition of the RSA Cryptosystem. Since the introduction of RSA, several variants have been presented. This variants differ from the original RSA-Scheme in that the values of some parameters are slightly

```
Algorithm 7: RSA Decryption
  Input: Private Key $d$ and ciphertext $c$.
  Output: Plaintext $m$ corresponding to ciphertext $c$.
  begin
      A (the receiver) should do the following:
      Step 1.Use the private key $d$ to recover $m = c^d \, mod \, N$.
  end
```

changed or in that there are some additional assumptions regarding these parameters.Throughout this chapter we will consider some of these variants. However, whenever we refer to RSA we will mean the Scheme and notation presented above unless otherwise stated.

In RSA Cryptosystem, the trapdoor one-way function is the function $m \rightarrow m^e$ $(mod \, N)$. Indeed, the above function can be easily computed but (as far as we know) cannot be efficiently inverted without the knowledge of the trapdoor d. However, if one knows the decryption exponent d, then one can recover the plaintext m as follows:
Since $e{\cdot}d \equiv 1 \, (mod \, \phi(N))$, there exists an integer k such that $ed = 1{+}k\phi(N)$.Consider the following two cases:

(a) $\gcd(m, p) = 1$. Then by Fermat's little theorem

$$m^{p-1} \equiv 1 \, (mod p).$$

If we raise both sides of this congruence to the power $k(q-1)$ and then multiply both sides by m we get

$$m^{1+k(p-1)(q-1)} \equiv m \, (mod p) \Rightarrow m^{ed} \equiv m \, (mod p).$$

(b) $\gcd(m, p) = p$. Then $m^{1+k(p-1)(q-1)} \equiv m \, (mod p)$ holds trivially as both sides are equivalent 0 mod p. Thus again $m^{ed} \equiv m \, (mod p)$.

Using the same arguments we can prove that

$$m^{ed} \equiv m \, (mod \, q).$$

Finally the fact that p, q are distinct primes (which means that $\gcd(p, q) = 1$), along with the Chinese Remainder Theorem, yield that

$$m^{ed} \equiv m \, (mod \, N).$$

and hence

$$c^d \equiv (m^e)^d \equiv m \, (mod \, N).$$

4.2 Computing $d \Leftrightarrow$ Factoring

In this section we present a positive application of lattices to the RSA Cryptosystem. By the term "positive" we mean an appplication that establishes the security of one RSA parameter. In particular, we present a result due to May [27] that establishes the deterministic polynomial time equivalence between computing the RSA secret key and Factoring.

While a successful attack against a cryptosystem is sufficient to prove that the cryptosystem is not secure, any number of unsuccessful attacks does not suffice to prove that the cryptosystem is in fact secure. How can we then establish that a cryptosystem is secure? In public-Key Cryptography, where the encryption process is based on an one-way function that is hard to invert, security could be established if we could prove the polynomial time equivalence between the problem of recovering the plaintext m from the ciphertext c without the knowledge of the trapdoor and a well-known hard problem P, believed to be computationally intractable.

It is not hard to see that RSA is directly related to the problem of factoring the modulus N which is considered to be hard. Indeed, once we recover p, q, we can compute $\phi(N) = (p-1)(q-1)$ and consequently decrypt any ciphertext c by computing the unique $d \in [0, \phi(N)]$ such that $ed \equiv 1 \pmod{\phi(N)}$. Thus, we could probably establish the security of RSA by proving that recovering the plaintext m from the ciphertext $c = m^e\, (mod\, N)$ and the public key, is polynomially time equivalent to factoring the modulus N. This is a very important open problem in Public-Key Cryptography.

Alternatively we can content ourselves with proving that recovering some secret information about RSA is equivalent to factoring. For example computing the value $\phi(N)$ is equivalent to factoring the modulus N, since we can both compute $\phi(N) = (p-1)(q-1)$ if we know p, q and the factorization of N if we know the value $\phi(N)$ by solving the system

$$\begin{aligned} &N = p \cdot q \\ &\phi(N) = N - (p+q) + 1. \end{aligned}$$

In 2004, May [27] proved that computing the RSA secret key d is deterministic polynomial time equivalent to factoring. This result establishes the satisfaction of a very fundamental requirement for a Public-Key Cryptosystem, namely the hardness of recovering the secret key from the public key. Indeed, the above result implies that an efficient [1] algorithm that recovers the secret key d from the public key e can be transformed to an efficient algorithm that factors N. This renders the existence of efficient algorithms that recover d impossible, provided that there is no efficient algorithm that factors N.

However, the above result does not provide any security for the public-key cryptosystem itself since there might be other ways to break the system without

[1]By the word efficient we usually mean algorithms that run in time polynomial in their input size.

computing the secret key d.

Previous Results. The problem of the polynomial time equivalence between computing d and factoring has been well studied in the past. Two of the most interesting previous results are:

- Existence of probabilistic polynomial time reduction between the above problems. A proof can be found in [37, pages 197-200] and in several other sources.
- Deterministic Polynomial Time equivalence under the Extended Riemann Hypothesis (ERH). The equivalence is directly established if we assume the validity of the ERH and a result based on a paper by Miller [32].

The presentation is separated into two parts. We first present May's result for balanced p, q and then a recent generalization due to Coron and May [15] for unbalanced p, q.

4.2.1 Balanced primes p, q

In his initial paper [27], May proved the equivalence between computing d and factoring N under the following two assumptions:

(a) $ed \leq N^2$ and

(b) p, q are of the same bitsize.

Assume wlog that $p < q$. Then the second assumption implies that

$$p < N^{1/2} < q < 2p < 2N^{1/2}$$

which gives the following inequalities

$$p + q < 3N^{1/2} \quad \text{and} \tag{4.1}$$

$$\phi(N) = N + 1 - (p + q) > \frac{N}{2}. \tag{4.2}$$

The last inequality is directly derived from $p + q < 3N^{1/2} \leq \frac{N}{2}$ (for $N \geq 36$).

In order to illustrate the underlying idea,we first give May's proof for a slightly weaker theorem, where we assume that $ed \leq N^{\frac{3}{2}}$.

Theorem 4.2.1
Let $N = pq$ be the RSA-modulus, where p and q are of the same bitsize. Suppose we know integers e, d such that $ed > 1$ and

$$ed \equiv 1 (mod\, \phi(N)), \quad ed \leq N^{\frac{3}{2}}.$$

Then N can be factored in time polynomial in its bitsize.

Proof. In the proof we use the following notation:

- $\lceil k \rceil$:ceiling of k.
- $\mathbb{Z}^*_{\phi(N)}$: Ring of the invertible integers mod $\phi(N)$.

In addition the relation $ed \equiv 1 \pmod{\phi(N)}$ gives that

$$ed = k\phi(N) + 1 \text{ for some } k \in \mathbb{N}.$$

We will now show how to compute k efficiently. We know that $k = \frac{ed-1}{\phi(N)}$. We define $\bar{k} = \frac{ed-1}{N}$ which is obviously an underestimation for k, that is $k \geq \lceil \bar{k} \rceil$. In addition we have that

$$\begin{aligned} k - \bar{k} &= \frac{ed-1}{\phi(N)} - \frac{ed-1}{N} = \frac{N(ed-1) - (N-p-q+1)(ed-1)}{\phi(N)N} \\ &= \frac{(ed-1)(p+q-1)}{\phi(N)N}. \end{aligned}$$

By inequalities 4.1 and 4.2 we get that $p+q-1 < 3N^{1/2}$ and $\frac{1}{\phi(N)N} \leq \frac{2}{N^2}$.Hence,

$$k - \bar{k} = \frac{(ed-1)(p+q-1)}{\phi(N)N} < 6N^{-3/2}(ed-1). \tag{4.3}$$

Since by hypothesis $ed \leq N^{\frac{3}{2}}$ we get that $k - \bar{k} < 6 \Rightarrow k - \lceil \bar{k} \rceil < 6$.
This means that we only have to try $\lceil \bar{k} \rceil + i$ for $i = 0, ..., 5$ to find the right k. For the right value of k, p, q can be recovered by the solution of the system

$$\begin{aligned} &N = p \cdot q \\ &N + 1 - \tfrac{ed-1}{k} = p + q. \end{aligned}$$

Obviously, in order to determine the correct value of k, we only need elementary arithmetic operations on integers of size $\log N$. The running time of the algorithm is apparently $\mathcal{O}(\log^2 N)$ which concludes the proof. □

In order to extend the above result to the case where $ed \leq N^2$, May uses Coppersmith's result for finding small solutions to bivariate integer equations presented in the previous chapter. Here we restate the theorem for convenience.

Theorem 4.2.2 (**Coppersmith's Theorem for Bivariate Integer Equations**) *Let $f(x, y)$ be an irreducible polynomial in two variables over $\mathbb{Z}$, of maximum degree δ in each variable separately. Let X, Y be upper bounds on the desired integer solution (x_0, y_0).Let W be the absolute value of the largest entry in the coefficient vector of $f(xX, yY)$,that is $W = \max_{i,j} |f_{i,j}|X^i Y^j$. If*

$$XY \leq W^{\frac{2}{3\delta}}$$

then in time polynomial in $\log W$ and 2^δ we can find all integer pairs (x_0, y_0) such that $f(x_0, y_0) = 0, |x_0| \leq X$ and $|y_0| \leq Y$.

May's main result is given by the following theorem

Theorem 4.2.3
Let $N = pq$ be the RSA-modulus, where p and q are of the same bitsize. Suppose we know integers e, d with $ed > 1$ and

$$ed \equiv 1 (mod\, \phi(N)), \quad ed \leq N^2.$$

Then N can be factored in time polynomial in its bitsize.

Proof. We again begin with the equation

$$ed = k \cdot \phi(N) + 1 \text{ for some } k \in \mathbb{N}. \tag{4.4}$$

We define again $\bar{k} = \frac{ed-1}{N}$ which underestimates k. Using inequality 4.3 from the proof of the previous theorem we obtain

$$k - \bar{k} < 6N^{-\frac{3}{2}}(ed-1) < 6N^{\frac{1}{2}}.$$

Apparently,the previous method cannot work since we would have to search for an exponentially to the bitsize of N number of possible values for k. May uses an alternative approach.

Let us denote $x = k - \lceil \bar{k} \rceil$. Then $\lceil \bar{k} \rceil$ is an approximation of the right value for k up to the additive error x. In addition inequality 4.1 gives

$$N - \phi(N) = p + q - 1 < 3N^{1/2}.$$

This means that $\phi(N)$ lies in the interval $[N - 3N^{\frac{1}{2}}, N]$. We divide the above interval into 6 subintervals of length $\frac{1}{2}N^{\frac{1}{2}}$ with centers $N - \frac{2i-1}{4}N^{1/2}$ for $i = 1, ..., 6$. For the correct i we have

$$|N - \frac{2i-1}{4}N^{\frac{1}{2}} - \phi(N)| \leq \frac{1}{4}N^{\frac{1}{2}}.$$

Let $g = \lceil \frac{2i-1}{4}N^{\frac{1}{2}} \rceil$ for the right i. Then

$$|N - g - \phi(N)| < \frac{1}{4}N^{1/2} + 1 \Rightarrow \phi(N) = N - g - y$$

for some unknown y with $|y| < \frac{1}{4}N^{\frac{1}{2}} + 1$. If we replace k and $\phi(N)$ in equation 4.4 we get

$$ed - 1 - (\lceil \bar{k} \rceil + x)(N - g - y) = 0. \tag{4.5}$$

We define now the bivariate integer polynomial :

$$f(x, y) = xy - (N - g)x + \lceil \bar{k} \rceil y - \lceil \bar{k} \rceil (N - g) + ed - 1.$$

Notice that f is exactly the reordering of the left side of equation 4.5.By construction, we know that $(x_0, y_0) = (k - \lceil \bar{k} \rceil, p + q - 1 - g)$ is a root of $f(x, y)$ over the integers.

We will use theorem 4.2.2 to show that the root (x_0, y_0) can be recovered in polynomial time. Let $X = 6N^{\frac{1}{2}}$ and $Y = \frac{1}{4}N^{\frac{1}{2}}+1$. Then $|x_0| \leq X$ and $|y_0| \leq Y$. Let W denote the l_∞ norm of the coefficient vector of $f(xX, yY)$. Then by inspection of the polynomial f we know that

$$\begin{aligned} W \geq (N - g)X &= (N - \lceil \frac{2i-1}{4} N^{\frac{1}{2}} \rceil) \cdot 6N^{\frac{1}{2}} \\ &= 6N^{\frac{3}{2}} - 6N^{\frac{1}{2}} \cdot \lceil \frac{2i-1}{4} N^{\frac{1}{2}} \rceil > 3N^{\frac{3}{2}} \end{aligned}$$

for sufficiently large N. Thus

$$\begin{aligned} XY = 6N^{\frac{1}{2}}(\frac{1}{4}N^{\frac{1}{2}} + 1) = \frac{3}{2}N + 6N^{\frac{1}{2}} < \\ < 2N < 3^{\frac{2}{3}}N = (3N^{\frac{3}{2}})^{\frac{2}{3}} < W^{\frac{2}{3}} = W^{\frac{2}{3\delta}}. \end{aligned}$$

All the inequalities are true for large N.

By Coppersmith's theorem we can find the root (x_0, y_0) in time polynomial in the bitsize of W and finally recover the factorization of N by the root $y_0 = p + q - 1 - g$.

In order to bound the running time of the above algorithm with respect to N, notice that a simple inspection of the polynomial $f(x, y)$ gives that $W \leq NX = 6N^{3/2}$. Since Coppersmith's approach gives results in time polynomial in $\log W$ and W is polynomially bounded by N, the running time of the algorithm is polynomial in $\log N$ too. This completes the proof. $\square$

Remark 4.2.4. Both previous results can be easily generalized for the case where $p + q \leq poly(logN)N^{\frac{1}{2}}$. Indeed

(a) For the case where $ed \leq N^{\frac{3}{2}}$ we only have to examine the values $\lceil \bar{k} \rceil + i$, for $i = 0, 1, ..., \lceil 2poly(logN) \rceil - 1$ (polynomially bounded by the bitsize of N.)

(b) For the case where $ed \leq N^2$ we just have to divide the interval $[N - poly(logN)N^{\frac{1}{2}}, N]$ into $\lceil 2poly(logN) \rceil$ subintervals and run the algorithm for each subinterval.

Remark 4.2.5. The above results can be summarized to the following interesting (from the cryptographic point of view) result.

Theorem 4.2.6
Let $N = pq$ be the RSA-modulus, where p and q are of the same bitsize. Furthermore let $e \in \mathbb{Z}^*_{\phi(N)}$ be an RSA public exponent. Suppose we have an algorithm that

on input (N, e) outputs in deterministic polynomial time the RSA secret exponent $d \in \mathbb{Z}^*_{\phi(N)}$ satisfying

$$ed = 1(mod\, \phi(N)).$$

Then N can be factored in deterministic polynomial time.

Notice that in the ordinary case (algorithm 5),in fact $e, d \in \mathbb{Z}^*_{\phi(N)}$. This strengthens the power of the result proved by May. Of course, as stated in remark 4.1.2, the encryption and decryption processes work even if $e, d \notin \mathbb{Z}^*_{\phi(N)}$.

4.2.2 Unbalanced primes p, q

Shortly after May's initial paper, Coron and May [15] revisited the above problem.They provided an alternative proof for theorem 4.2.3 using a variant of Coppersmith's technique for finding small solutions to univariate modular equations (instead of bivariate integer equations).

Interestingly, Coron and May [15] proved that the equivalence between factoring and computing the secret key d is still valid even if the requirement that p, q are balanced is removed. In fact, they proved that factoring N given (e, d) becomes easier when the prime factors are unbalanced.Their technique is similar to the technique introduced by Durfee and Nguyen [18] in which two separate variables x and y are used for the primes p and q respectively and each occurence of $x \cdot y$ is replaced by N.

More specifically,they proved the following theorem.

Theorem 4.2.7
Let β and $0 < \delta \leq \frac{1}{2}$ be real values, such that $2\beta\delta(1-\delta) \leq 1$.Let $N = pq$, where p, q are primes such that $p < N^\delta$ and $q < 2N^{1-\delta}$.Let e, d be such that $e \cdot d \equiv 1$ mod $\phi(N)$, and $0 < e \cdot d \leq N^\beta$. Then given (N, e, d) one can recover the factorization of N in deterministic polynomial time.

Remark 4.2.8. The factorization of N is easier when p, q are unbalanced in that the condition for the product $e \cdot d$ becomes weaker.Consider for example that $p < N^{\frac{1}{4}}$. Plugging the value $\delta = \frac{1}{4}$ in the inequality $2\beta\delta(1-\delta) \leq 1$ yields $\beta \leq \frac{8}{3}$. This means that the proof of equivalence between computing d and factoring N can now tolerate values of the product $e \cdot d$ up to $N^{\frac{8}{3}}$ (instead of N^2). Of course letting $\delta = \frac{1}{2}$ (balanced p, q) we get the same result as in the previous subsection $(e \cdot d \leq N^2)$.

Chapter 5

Lattice-Based Attacks on RSA

Since its initial publication, the RSA Cryptosystem has been analyzed for vulnerabilities by many researchers. However, none of the attempted attacks has proven devastating. The attacks mostly illustrate the dangers of improper use of RSA. Boneh [4] presents a thorough overview of the most successful attacks mounted against RSA the first twently years after its publication.

The development of lattice theory and the invention of LLL algorithm has motivated a number of lattice attacks on RSA Cryptosystem. In this section we will only consider attacks on RSA that are related to lattice methods and present the underlying ideas.

It is important to note that none of the attacks described below reveals any flaw to the RSA Cryptosystem. Despite the large number of attacks addressed to RSA, none of them has managed to break RSA in its general setting. All of the attacks described here utilize certain flaws induced by insecure choises of the parameters rather than inherent flaws of the Cryptosystem itself. In particular, we will study the following insecure choises of parameters that are susceptible to lattice attacks.

(a) Section 5.1: Low public exponent e.

(b) Section 5.2: Exposure of a fraction of the most (or less) significant bits of one of the primes p or q.

(c) Section 5.3: Low private exponent d.

(d) Section 5.4: Partial exposure of the private exponent d.

In all cases, we assume that the communication channel between entities A and B is insecure, that is an eavesdropper E (usually referred to as Eve) has access to the full ciphertext c transmitted through the channel. Otherwise, he would not be able to obtain the plaintext m even if he knew the private exponent d. The aim of a secure public-key Cryptosystem is to make the recovery of m infeasible even if the ciphertext c is competely known.

5.1 Low Public Exponent Attacks

In many practical applications, the encryption process is performed by some limited device, such as a smart card. In such cases, raising a plaintext m to a high power might be costly in terms of power consumption or time. In an attempt to simplify the encryption process, one might be tempted to use a small public exponent e. A typical value for the exponent is $e = 3$, which means that the encryption process only involves raising a number to the power 3, which can be trivially done using two multiplications.In this section we argue that the use of small exponents can induce serious threats to RSA's security.

The attacks described below take advantage of the fact that the public exponent e used for encryption is relatively small. Unlike the attacks described in the next sections, the attacks described here can only be used to recover a given plaintext and do not expose the private key.

5.1.1 Stereotyped Messages

Consider the following scenario:
The eavesdropper does not know the full plaintext message m but knows a part of it. That is $m = M + x$ where M is known and thus it suffices to recover x in order to fully recover the initial message. For a realistic perspective suppose that the director of a large bank renews the secret key (subsequently used for symmetric encryption) every day and sends it to the branches of the bank.The daily message looks like " Good morning to everyone. Today's secret key is ...". Of course the director is clever enough to encrypt the above message before sending it. Suppose that the Cryptosystem used for the encryption of the above message is RSA. Then the eavesdropper is confronted with the following challenge:

Given a ciphertext $c \equiv (M + x)^e \, (mod\, N)$ recover the unknown part x of the plaintext.

In the general setting (where the secret exponent e is arbitrary) E cannot do much. However, in the specific case we are talking about, e is small. In order to recover x, E forms the following polynomial

$$f(x) = (M + x)^e - c \equiv 0 \, (mod\, N).$$

Now recall Coppersmith's main result for univariate modular polynomial equations proved in the previous chapter.

Theorem 5.1.1 (Coppersmith Theorem for Univariate Modular Equations)
Let N be an integer of unknown factorization. Furthermore, let $f(x)$ be a univariate, monic polynomial of degree δ. Then we can find all solutions x_0 for the equation $f(x) \equiv 0 (mod\, N)$ with

$$|x_0| \leq N^{\frac{1}{\delta}}$$

in time polynomial in $(\log N, \delta)$.

This means that E can recover x efficiently as long as $|x| < N^{\frac{1}{e}}$. But this condition may be easily satisfied in practice when RSA is used with small exponent e (for instance with the frequently used choice $e = 3$). Indeed, notice that the encryption process described in algorithm 6 implies that the integer $m = M + x$ that corresponds to the initial message is an integer in the range $[0, N-1]$.This means that $x = m - M < N - M$ is likely to satisfy $|x| < N^{\frac{1}{e}}$.

Remark 5.1.2. The above scenario , where e is small, is in fact rather realistic in practice where symmetric cryptosystems often need keys of length at most 80 bits (this length corresponds to the length of x in the modular equation).Thus if $e = 3$ and the known part M of the message is more than 160 bits (20 8-bit ASCII characters), an eavesdropper can efficiently recover x using Coppersmith's technique for univariate modular equations.

Remark 5.1.3. It is not difficult to observe that a similar result holds in the case where the unknown part x is somewhere in the middle of the message. For instance assume that the initial message is of the form $m = M + x2^k + M'$ where x begins at the $k + 1^{st}$ least significant bit. Then the polynomial the roots of whose need to be recovered is $f(x) = (M + x2^k + M')^e - c$. Theorem 5.1.1 cannot be directly applied because f is not monic. However since N is odd, $\gcd(2^{ke}, N) = 1$ and thus the polynomial $2^{-ke} f(x) \bmod N$ exists and is monic. We can then apply 5.1.1 to $2^{-ke} f(x) \bmod N$ and recover x.

5.1.2 Hastad's Broadcast Attack

Suppose now that a sender wants to send a message m to a number of parties $P_1, P_2, ..., P_k$. Each party P_i has its own RSA public key (N_i, e_i). Assume that $m < \min_i N_i$. In order to send m to all parties, the sender encrypts m and sends $m^{e_i}\,(mod\, N_i)$ to each party P_i. We illustrate below how an eavesdropper E can recover the initial message m given the k ciphertexts c_i. For simplicity we assume that $e_i = 3$ and that the recepients are 3 ($k = 3$).The extension to larger e_i (and corresponding larger k) is then straightforward.

Suppose that E obtains c_1, c_2, c_3 where

$$c_1 = m^3\, mod\, N_1, \quad c_2 = m^3\, mod\, N_2, \quad c_3 = m^3\, mod\, N_3.$$

We assume that $\gcd(N_i, N_j) = 1$ for $i \neq j$. Otherwise a nontrivial divisor of N_i can be found and then recovering m is easy. Thus we can apply CRT [1] and find a c' such that $c' \equiv m^3\,(mod\, N_1N_2N_3)$.CRT along with the condition $\gcd(N_i, N_j) = 1$ guarantees that c' is unique mod $N_1N_2N_3$. In addition since $m < \min\{N_1, N_2, N_3\}$ we have that $m^3 < N_1N_2N_3$. This gives that $m^3 = c'$ over the integers and E can

[1]Chinese Remainder Theorem

then recover m by simply computing the real cube root (and not the modular cube root) of c'.

It is not difficult to see that a similar attack can work for all small (common) public exponents e and k parties as long as $k \geq e$.

It seems that we can avoid the above attack by never sending the same message to more than one person. For instance, consider the following solution: Each person, as part of his public key, has some unique id.Instead of encrypting m the sender encrypts $m + 2^k \cdot id$ where k is the length of the message m in bits and id is the id of the recipient. In this way, the sender never sends the same message to more than one persons.

Hastad [21] showed that the above padding is also insecure. In fact, he introduced a much more general attack. Assume that the public key of each recipient is of the form (N, g) where g is some polynomial in m. The encryption of a message is given by $g(m) \, mod \, N$. For example in the case above $g(m) = (m + 2^k id)^3$. The following theorem is a stronger version of Hastad's original result.

Theorem 5.1.4 (**Hastad**)
Let $N_1, N_2, ..., N_k$ be pairwise relatively prime integers and let $N_{min} = \min_i N_i$. Let $g_i \in \mathbb{Z}_{N_i}[x]$ be k polynomials of maximum degree d. Suppose that there exists a unique $m < N_{min}$ such that $g_i(m) = c_i \, (mod \, N_i)$ for all $i = 1, 2..., k$. Then, if $k \geq d$, one can efficiently find m given $(N_i, g_i, c_i)_{i=1}^k$.

Proof. Define $h_i = g_i - c_i$ for $1 \leq i \leq k$. We are then looking for m such that $h_i(m) = 0 \, (mod \, N_i)$ for $i = 1, 2, ..., k$. Assume wlog [2] that all h_i are monic. If not we can multiply the polynomials with the inverse mod N_i of their leading coefficients in order to make them monic.(If there is no such inverse then we can again factor N_i and things get easier anyway). In addition we can multiply (if necessary) each polynomial h_i by x^j to make them all have degree d.

We now use the CRT to combine the polynomials h_i into a single polynomial h. In particular,we define

$$h(x) = \sum_{i=1}^{k} T_i h_i(x) \ (mod \, N)$$

where $N = N_1 N_2 \cdots N_k$ and

$$T_i = \begin{cases} 1 \, mod \, N_j & \text{if } i = j \\ 0 \, mod \, N_j & \text{if } i \neq j \end{cases}$$

can be obtained using CRT to the above linear system of modular equations. The polynomial h as defined above has the following properties:

1. It has degree d as the sum of polynomials of degree d.

[2]without loss of generality

2. It is monic since its x^d coefficient is 1 modulo any N_i. For that notice that $\sum_{i=1}^{k} T_i = 1\,(mod\,N_i)$ and by applying CRT we get that $\sum_{i=1}^{k} T_i = 1\,(mod\,N)$.

3. $h(m) \equiv 0\,(mod\,N)$.

Remember here that $m < \min_i N_i$ and $d \leq k$ which means that

$$m^d \leq m^k < \prod_{i=1}^{k} N_i = N \Rightarrow m < N^{\frac{1}{d}}.$$

This means that we can construct a polynomial h of degree d such that $h(m) \equiv 0\,(mod\,N)$ and $m < N^{\frac{1}{d}}$. We can now apply theorem 5.1.1 in order to efficiently find m. This completes the proof. □

Remark 5.1.5. Hastad's original theorem is weaker than the one stated above. Hastad required $\frac{d(d+1)}{2}$ rather than d polynomials.The initial proof given by Hastad is similar to the proof given by Coppersmith for univariate modular equations. However, in contrast to Coppersmith, Hastad does not use powers of polynomials g grater than 1 which leads to smaller lattices and consequently weaker bounds.

Remark 5.1.6. Despite the various attacks, low public exponent RSA is still considered secure when used carefully. The current wisdom says that one should use a moderate public exponent, let's say $e = 2^{16} + 1$ and pad the message with some random bits.

5.1.3 Random Padding to Messages

The following analysis illustrates an alternative low public exponent attack. In this setting, there is only one recepient but more than one (related) messages sent to him by the sender. The attack described below was first motivated by a result due to Franklin and Reiter [20]. More specifcally, here we describe only a simple instance of their attack in order to illustrate the underlying idea.

Consider that the sender sends two encrypted messages. The plaintexts are m and m' and the corresponding ciphertexts c and c' respectively. Suppose now that the above two messages m, m' satisfy a known affine relation

$$m' = m + t$$

where t is known. We will show that an eavesdropper E can recover the message m (and thus m' too) if he is given the ciphertexts c, c'.Indeed, the following modular equations hold

$$\begin{aligned} c &\equiv m^3 && (mod\,N) \\ c' &\equiv m'^3 \equiv m^3 + 3m^2t + 3mt^2 + t^3 && (mod\,N) \end{aligned}$$

Then it is not difficult to see that m is given by

$$m = t(c' + 2c - t^3)(c' - c + 2t^3)^{-1}\,(mod\,N)$$

where $(c' - c + 2t^3)^{-1}$ denotes the inverse of $(c' - c + 2t^3)$ modulo N.

Remark 5.1.7. Notice that the case where the two messages are of the form $m_1 = m + r_1$ and $m_2 = m + r_2$ where r_1, r_2 are known is completely equivalent as we can express m_2 in terms of m_1 as follows: $m_2 = m_1 + t$ where $t := r_2 - r_1$.

Remark 5.1.8. In their original paper, Franklin and Reiter [20] proved a slightly more genaral result. In the instance they presented, there are again two messages m_1 and m_2 which are related by the more general affine relation

$$m_2 = \alpha m_1 + \beta$$

where α, β are known parameters. The public exponent is again $e = 3$ and the messages required in order for an eavesdropper to be able to recover m_1, m_2 are $k = 2$.Of course both messages are encrypted with the same public key pair (N, e).

In 1996 , Coppersmith , Franklin, Patarin and Reiter [13] generalized the above attack. The attacks they present, enable the recovery of plaintext messages from their ciphertexts and a known polynomial relationship among the messages, provided that the ciphertexts were created using the same RSA public key with low encryption exponent. That is, they presented a technique which can recover the initial k plaintexts in the general case where the relation between them is a polynomial of degree δ and not a linear relation as in the initial presentation by Franklin and Reiter. In addition, they showed how to handle cases where e is still small but does not necessarily equal 3. All the details for the above generalization can be found in [13].

The attacks presented by Franklin and Reiter and by Coppersmith, Franklin, Patarin and Reiter seem a little artificial in that they presume that the eavesdropper knows the affine relation among the messages. In addition, the assumption that the messages sent by the sender are related is a little nebulous.

Let us return to the initial setting where $m' = m + t$ and $e = 3$.A natural question to ask is what happens when t is not known. Coppersmith [10] showed that we can still recover m as long as t is short enough. The above attack is of great practical importance since someone could hope to remove the flaws described previously by subjecting each message to random padding before encrypting it with RSA.This technique was proven insufficient by Coppersmith for all cases where the padding was short compared to the modulus N.

Assume for example that the sender shifts the initial message M by k bits before sending it and adds a random k-bit quantity T to get the corresponding plaintext m. Suppose then that the same message is encrypted twice, but with different random padding $T' = T + t$. This corresponds to a plaintext $m' = m + t$.Then the two ciphertexts c, c' are given by

$$\begin{array}{lll} c \equiv m^3 & \equiv (2^k M + T)^3 & (mod\, N) \\ c' \equiv m'^3 & \equiv (2^k M + T')^3 \equiv (m+t)^3 & (mod\, N). \end{array} \tag{5.1}$$

A straightforward but tedious computation shows that if we eliminate m by taking the resultant of the polynomials $m^3 - c$ and $(m+t)^3 - c'$ we get

$$\begin{array}{l} res_m(m^3 - c, (m+t)^3 - c') = \\ = t^9 + (3c - 3c')t^6 + (3c^2 + 21cc' + 3c'^2)t^3 + (c - c')^3 \equiv 0\, (mod\, N). \end{array}$$

This means that by equation 5.1 we can construct a univariate polynomial in t of degree 9 (mod N). We can now recover t (and consequently m using the methods described above) using Coppersmith's technique for univariate modular equations (theorem 5.1.1) as long as $|t| < N^{\frac{1}{9}}$. This means that if the message is subject to random padding of length less than 1/9 the length of N, and then encrypted with an exponent $e = 3$, multiple encryptions of the same message will reveal it.

Remark 5.1.9. Notice that the same attack can work just as well if the padding goes in the high order bits or even in the middle. In that case we only have to divide each ciphertext by the appropriate power of 2, in order to divide each plaintext by another power of 2, to move the random padding to the low order bits.

5.2 Lattice-Based Factoring Techniques

In this section we consider attacks where an eavesdropper E tries to break RSA by factoring the modulus N. Notice first that once the eavesdropper manages to factor N, he can then decrypt any message.Indeed, once the primes p, q are known, E can compute $\phi(N)$ and then find the unique secret exponent $d \in [1, \phi(N)]$ such that $ed \equiv 1\, (mod\, \phi(N))$. Using d he can decrypt any ciphertext c just like the legitimate receiver. We first present lattice-based factoring techniques for the original RSA-Scheme where the modulus is of the form $N = pq$. Afterwards, we present the extension of these techniques to recently proposed RSA variants where $N = p^r q$.

It is important to note here that the attacks for both cases of modulus N are much stronger than the low public exponent attacks presented in section 5.1 in that we can decrypt any message once we manage to factor the modulus N. In the previous section, the decryption of a message m did not reveal anything for another message m'. If subsequent messages were selected with care by the sender, then the eavesdropper would not be able to recover the plaintexts. However, in the attacks described below, the communication between the sender and the legitimate receiver is completely broken once the modulus N is factored. In order to re-establish a secure communication between him and the sender, the receiver has to choose a new pair of keys (N, e) and d.

5.2.1 Factoring RSA-moduli $N = pq$ by knowing half of the bits of p

Unlike the previous section, the attacker in current scenarios may get parts of one of the factors p, q instead of parts of the message m.We will present a method which enables us to factor the modulus N in time polynomial in its bitsize provided that we know half of the bits of p and p, q are of the same bitsize.In his initial proof , Coppersmith [11] used his results for finding small solutions to bivariate integer equations. More specifically, Coppersmith proved the following theorem (the proof can be found in [11].)

Theorem 5.2.1 (**Coppersmith, Factoring N with high bits known**)
Let $N = p \cdot q$ where p, q primes. Then given the high-order $(\frac{1}{4}\log_2 N)$ bits of p, one can factor N in time polynomial in its bitsize.

Here we present Coppersmith's results in a slightly more general form than the original. This generalization will simplify the subsequent analysis. Here we assume wlog that $p > q$.For simplicity, we state again the Coppersmith's generalized theorem for solving univariate modular equations proved in a previous chapter.

Theorem 5.2.2 (**Coppersmith Generalized Theorem for Univariate Modular Equations**)
Let N be an integer of unknown factorization, which has a divisor $b \geq N^{\beta}$. Furthermore, let $f(x)$ be a univariate, monic polynomial of degree δ and c_N be a function that is upper-bounded by a polynomial in $\log N$. Then we can find all solutions x_0 for the equation $f(x) \equiv 0 (mod\, b)$ with

$$|x_0| \leq c_N N^{\frac{\beta^2}{\delta}}$$

in time polynomial in $(\log N, \delta)$.

We can now prove the following theorem.

Theorem 5.2.3
Let $N = kp$ with $p > q$. Furthermore, let k be an (unknown) integer that is not a multiple of q. Suppose that we know an approximation $\tilde{p}$ of kp such that

$$|kp - \tilde{p}| \leq 2N^{\frac{1}{4}}.$$

Then we can find the factorization of N in time polynomial in $\log N$.

Proof. Define first the polynomial

$$f_p(x) := x + \tilde{p},$$

where the index p denotes that we are interested in finding roots of the above polynomial modulo p. Obviously $x_0 = kp - \tilde{p}$ is a root of $f_p(x) \bmod p$. In addition, the above polynomial has degree $\delta = 1$ and we know that a lower bound for p is $p \geq N^{\frac{1}{2}}$ since we have assumed that $p > q$.We can then apply 5.2.2 with $b = p, \beta = \frac{1}{2}, \delta = 1$ and $c_N = 2$ and get the root x_0 in time polynomial in $\log N$. But the knowledge of $x_0, \tilde{p}$ yield kp. Since by assumption k is not a multiple of q, the computation of $\gcd(N, kp)$ yields p and thus the factorization of N. □

Notice that the above theorem implies theorem 5.2.1 when p, q are of the same bitsize. To see that, recall that the fact that p, q are roughly of the same size implies that $p \approx N^{\frac{1}{2}}$. This means that p has approximately $\frac{n}{2}$ bits where $n = \log_2 N$ is the bitsize of the modulus N. Suppose now that we know the $(\frac{1}{4}n)$ most significant bits of p. This means that we know half of the most significant bits of p. If we write $p = p_0 2^{\frac{n}{4}} + x$ then p_0 represents half of the most significant bits of p and $x \leq 2^{\frac{n}{4}}$. This means that we know p_0 such that

$$|p - p_0 2^{\frac{n}{4}}| = |x| \leq 2^{\frac{n}{4}} = N^{\frac{1}{4}}.$$

Theorem 5.2.3 with $k = 1, \tilde{p} = p_0 2^{\frac{n}{4}}$ then says that p can be recovered in polynomial time in $\log_2 N$.

Remark 5.2.4. Coppersmith in theorem 5.2.1 does not require that p, q are of the same bitsize. We can extend the approach used above in order to tolerate unbalanced p, q as follows:Suppose that $p \geq N^{\beta}$ and that its bitsize is approximately $\beta \cdot n$ where $\beta \geq \frac{1}{2}$. Then we can prove thorem 5.2.3 in a completely analogous way where the bound now becomes

$$|kp - \tilde{p}| \leq N^{\beta^2} \approx p^{\beta}.$$

This means that we can factor N if we know a fraction of $1 - \beta$ of the most significant bits of p or, since the bitsize of p is $\beta \cdot n$, it suffices to know the $(1-\beta)\beta n$ most significant bits of p. But for $\frac{1}{2} \leq \beta < 1$, $(1-\beta)\beta$ is a decreasing function of β with a local maximum $\frac{1}{4}$ for $\beta = \frac{1}{2}$. This means that the method described above is even more efficient when p, q are unbalanced.

Modifying theorem 5.2.3, we can obtain a similar result for the case where we know the less (instead of the most) significant bits of p.

Theorem 5.2.5
Let $N = pq$ where p, q are of the same bitsize with $p > q$. Suppose we know p_0 and M satisfying

$$p_0 = p \, mod M \quad and \quad M \geq N^{\frac{1}{4}}.$$

Then one can find the factorization of N in time polynomial in $\log N$.

Proof. Define the univariate polynomial

$$f_p(x) := xM + p_0.$$

Since $p_0 = p \, mod \, M$, the term $x_0 = \frac{p-p_0}{M}$ is an integer.In addition x_0 is a root of $f_p(x)$ modulo p since $f_p(x_0) = p$. Let M^{-1} be the inverse of M modulo N. (If such an inverse does not exist then $\gcd(M, N)$ yields the factorization of N.)Compute now the polynomial

$$f'_p(x) := M^{-1} f_p(x) \, mod \, N$$

which is monic and has the same root $x_0 = \frac{p-p_0}{M}$ modulo p. Since p, q are of the same bitsize, we have

$$q < \sqrt{N} < p < 2\sqrt{N}$$

which along with the fact that $M \geq N^{\frac{1}{4}}$ gives that

$$|x_0| = \frac{p - p_0}{M} \leq 2N^{\frac{1}{4}}.$$

By theorem 5.2.2 with parameter values $\beta = \frac{1}{2}, \delta = 1$ and $c_N = 2$, x_0 (and consequently p) can be found in time polynomial in $\log N$ which completes the proof. □

Remark 5.2.6. The connection between the above theorem and factoring with half of the LSB of p known is obvious if we let $M = 2^{\frac{n}{4}}$, where $n = \log_2 N$ the bitsize of N.Then,since p, q are of the same bitsize which means that the bitsize of p is approximately $\frac{n}{2}$, p_0 represents half of the LSB of p.

Remark 5.2.7. Analogous to the cases of half of the LSBs or MSBs of p is the case where we know an amount of half of the bits for any intermidiate consecutive bits.

5.2.2 Extension to moduli of the form $N = p^r q$

The main advantage of the method presented previously is that it can be extended to the case where the modulus is of the form $N = p^r q$.The latter form of the modulus N has appeared in recently proposed Cryptographic Schemes. In such schemes the RSA decryption process can be performed significantly faster than in the typical case where $N = pq$.

In this subsection, we will study in brief the extension of the approach followed in the previous subsection to $N = p^r q$ for the case where p, q are of the same bitsize. In that case, for the same bitsize of N, p, q have smaller bitsize than the usual case which means that the performance is improved.

In 1999, Boneh , Durfee and Howgrave-Graham [7] showed that schemes with moduli $N = p^r q$ are more susceptible to attacks that leak bits of p than the original RSA-Scheme. Their result generalizes theorem 5.2.3 to moduli of the form $N = p^r q$. We present below a more general form of the theorem proved by Boneh, Durfee and Howgrave-Graham.

Theorem 5.2.8 (**BDH,Factoring $N = p^r q$ with high bits known**)
Let $N = p^r q$, where r is a known constant and p, q are of the same bitsize. Let k be an (unknown) integer that is not a multiple of $p^{r-1}q$. Suppose we know an integer $\tilde{p}$ such that

$$|kp - \tilde{p}| \leq N^{\frac{r}{(r+1)^2}}.$$

Then N can be factored in time polynomial in its bitsize.

Proof. We again define the univariate monic polynomial

$$f_{p^r}(x) := (x + \tilde{p})^r.$$

Consider now the integer $x_0 = kp - \tilde{p}$. Then by hypothesis $|x_0| \leq N^{\frac{r}{(r+1)^2}}$. In addition, $f_{p^r}(x_0) = (kp)^r \equiv 0\,(mod\,p^r)$ which means that x_0 is a root of $f_{p^r}(x)$ modulo p^r.

The degree of $f_{p^r}(x)$ is $\delta = r$. In addition, since p, q are of the same bitsize, we know that $p > \frac{1}{2}q$ which gives that $p^{r+1} = \frac{Np}{q} > \frac{1}{2}N$. This implies that

$$p^r > (\frac{1}{2}N)^{\frac{r}{r+1}} > \frac{1}{2}N^{\frac{r}{r+1}}.$$

Define now $\beta := \frac{r}{r+1} - \frac{1}{\log N}$. Then

$$2N^{\frac{\beta^2}{\delta}} = 2N^{\frac{r}{(r+1)^2} + \frac{1}{r \log^2 N} - \frac{2}{(r+1)\log N}} \geq 2N^{\frac{r}{(r+1)^2} - \frac{1}{\log N}} = N^{\frac{r}{(r+1)^2}}$$

where the last equality is true since the logarithm base is 2 and thus $N^{\frac{1}{\log N}} = 2$. This means that theorem 5.2.2 with parameters $\beta := \frac{r}{r+1} - \frac{1}{\log N}, \delta = r, c_N = 2$ and $b = p^r$ is applicable and thus we can find x_0 in polynomial time.

It only remains to prove that $x_0 = kp - \tilde{p}$ yields the factorization of N. By x_0 we directly obtain kp which is not a multiple of N (since k is not a multiple of $p^{r-1}q$). This means that $\gcd(N, kp)$ is either of the form p^i or of the form qp^j for some integers $i \leq r, j < r$. The first case can yield p if we guess i (notice that the number of guesses required is polynomially bounded by the bitsize of N) and then compute the ith root of p^i. The second case can be reduced to the first case since $\frac{N}{qp^j} = p^{r-j}$ and p can be recovered by guessing $r - j$. This completes the proof. □

Remark 5.2.9. The implications of the above theorem are rather intereseting. Consider the case where $k = 1$. Since p, q are assumed to have the same bitsize, N is of size roughly p^{r+1}. This means that the condition of the previous theorem is equivalent to

$$|p - \tilde{p}| \leq p^{\frac{r}{(r+1)}}.$$

This means that a fraction of $\frac{1}{r+1}$ of the bits of p are sufficient to factor N.Notice that if $r = 1$ we obtain the result proved in the previous subsection.If N has 1000 bits, we need 250 bits of p (which has roughly 500 bits) if $r = 1$ while only 111 bits of p (whose bitsize is roughly 333) are sufficient to factor N in the case where $r = 2$. The warning is clear. The larger the exponent r, the more susceptible RSA-Schemes with modulus $N = p^r q$ are to such type of attacks. Boneh, Durfee and Howgrave-Graham [7] showed that if p, q have the same bitsize and $r = \epsilon \log p$ for a fixed constant $\epsilon > 0$, then we can factor the modulus $N = p^r q$ in polynomial time without any information about the bits of p.

5.3 Low Private Exponent Attacks

A serious drawback of RSA is its efficiency. A normal RSA decryption/signature generation requires time $\Theta(\log d \log^2 N)$. Selecting a small value for the secret exponent d can significantly increase the speed for the normal RSA decryption/signature processes. However, recent attacks against RSA, show that secret private exponents should be handled with care as they may threaten RSA's security.

In this section we present some attacks mounted against RSA instances with small secret exponents d. The attacks are somehow presented in a chronological order. Apart from attacks against the common RSA scheme, we also discuss attacks against RSA variants (for instance RSA schemes with modulus $N = p^r q$) or against schemes where the decryption process uses two small values (d_p, d_q) [3] related to d.

Here,like in section 5.2, the attacks for both cases of modulus N are much stronger than the low public exponent attacks presented in section 5.1. All of the attacks described below manage to factor N and totally expose the RSA Cryptosystem. In order to re-establish a secure communication between him and the possible senders, the legitimate receiver has to choose a new pair of keys (N, e) and d.

5.3.1 Wiener Attack

The first result showing that using small secret exponent can pose serious threats to RSA's security is due to Wiener [38]. In his paper, Wiener showed that a value of d less than $\frac{1}{3}N^{\frac{1}{4}}$ leads to a polynomial time attack on the RSA cryptosystem. More specifically, Wiener's results are summarized in the following theorem.

Theorem 5.3.1 (**Wiener**)
Let $N = pq$ with $q < p < 2p$. Let $d < \frac{1}{3}N^{\frac{1}{4}}$. Given (N, e) with $ed \equiv 1$ mod $\phi(N)$, then an eavesdropper can efficiently recover d.

His attack was based on continued fractions and did not use lattice techniques, that's why we just mention his contribution here. More specifically, he showed that d is the denominator of some convergent of the continued fraction expansion

[3]This values are known as CRT exponents.

of a number. The efficiency of Wiener's attack can be enervated if a large public exponent e is used. As the public exponent e gets larger the attack becomes less effective and cannot work at all if $e > N^{\frac{3}{2}}$.

5.3.2 Boneh & Durfee (BD) Small Inverse Attack

In 1999, Boneh and Durfee [5] presented the first substantial improvement over Wiener's bound. Their attack can (heuristically) recover the primes p, q in polynomial time provided that $d \leq N^{0.292}$. Their result is heuristic since it is based on Coppersmith's technique for finding small solutions to bivariate modular polynomial equations. However, their attack seems to work very well in practice. Below we present the main ideas of their approach.

Consider the normal RSA Scheme where p, q are balanced and assume for simplicity that $\gcd(p-1, q-1) = 2$ (similar analysis can be followed for the more general case).The keys e, d then satisfy $e \cdot d \equiv 1 \bmod \frac{\phi(N)}{2}$, where always $\phi(N) = N - p - q + 1$. This implies that there exists an integer k such that

$$ed + k\left(\frac{N+1}{2} - \frac{p+q}{2}\right) = 1. \tag{5.2}$$

Setting $s = -\frac{p+q}{2}$ and $A = \frac{N+1}{2}$ we get

$$k(A+s) \equiv 1\,(mod\,e).$$

Let $e = N^{\alpha}$ be of the same order of magnitude as N (and therefore α is close to 1).Boneh and Durfee actually show that values of α much smaller than 1 lead to even stronger results.Suppose now that d satisfies $d < N^{\delta}$. The goal is to push δ for which the factorization of N can be performed in polynomial time, to values as large as possible. Equation 5.2 imply that

$$|k| < \frac{2de}{\phi(N)} \leq \frac{3de}{N} < 3e^{1+\frac{\delta-1}{\alpha}}$$

where for the second inequality we have used that $\frac{\phi(N)}{N} > \frac{2}{3}$ and for the last that $e = N^{\alpha} \Rightarrow N = e^{\frac{1}{\alpha}}$.

Furthermore, since p, q are balanced, we know that $p, q < 2\sqrt{N}$ which gives

$$|s| < 2N^{\frac{1}{2}} = 2e^{\frac{1}{2\alpha}}.$$

If we take $\alpha \approx 1$ and ignore the small constants (which are negligible compared to the other terms appearing in the above inequalities) we end up with the following problem

Definition 5.3.2 (**Small Inverse Problem (SIP)**)
Given a polynomial $f(x, y) = x(A+y) - 1$, find (x_0, y_0) satisfying

$$f(x_0, y_0) \equiv 0\,(mod\,e) \quad where \quad |x_0| < e^{\delta} \text{ and } |y_0| < e^{0.5}.$$

Prooving the case $d < N^{0.284}$

By construction, we know that (k, s) is a small solution of the $f(x, y) = x(A + y) - 1 \equiv 0\,(mod\,e)$ satisfying $|k| < e^{\delta}$ and $|s| < e^{0.5}$. Thus, if we manage to solve the SIP for this instance, we will get $s = -\frac{p+q}{2}$ and consequently the factorization of N. The goal from now on is to recover the values of δ for which the roots (x_0, y_0) with $|x_0| < e^{\delta}, |y_0| < e^{0.5}$ can be recovered in polynomial time.Of course, we would like to prove that this modular equation is efficiently solvable for values of δ that are as large as possible.

Boneh and Durfee first transform the modular equation into an equation over the integers using Howgrave-Graham's Lemma for the bivariate case. We restate the bivariate version of the lemma for convenience.

Lemma 5.3.3 (Howgrave-Graham for Bivariate Integer Polynomials). Let $h(x, y) \in \mathbb{Z}[x, y]$ be a polynomial in 2 variables with at most ω monomials and let m be a positive integer. Suppose in addition that

1. $h(x_0, y_0) \equiv 0(mod\,e^m)$ where $|x_0| < X$ and $|y_0| < Y$, and
2. $\|h(xX, yY)\| \le \frac{e^m}{\sqrt{\omega}}$.

Then $h(x_0, y_0) = 0$ holds over the integers.

Next, for a given positive integer m they define the polynomials

$$g_{i,k}(x, y) := x^i f^k(x, y) e^{m-k} \qquad \text{(x-shift polynomials)}$$
$$h_{j,k}(x, y) := y^j f^k(x, y) e^{m-k} \qquad \text{(y-shift polynomials)}$$

Notice that (x_0, y_0) is a root of all these polynomials modulo e^m. In order to find a low-norm integer linear combination of the polynomials $g_{i,k}(xX, yY)$ and $h_{j,k}(xX, yY)$ and then apply lemma 5.3.3, Boneh and Durfee build a lattice spanned by the coefficient vectors of the polynomials $g_{i,k}$ and $h_{j,k}$ for certain parameters i, j and k. For each $k = 0, 1, ..., m$ they use the x-shifts $g_{i,k}(xX, yY)$ for $i = 0, 1, ..., m - k$. Additionally, they use the y-shifts $h_{j,k}(xX, yY)$ for $j = 0, 1, ..., t$ for some parameter t to be optimized later. A convenient ordering of the coefficient vectors renders a lower triangular matrix.

Let $\mathcal{L}_{BD}$ denote the lattice constructed by Boneh and Durfee and B_{BD} the corresponding basis.We will use the notation $B_{BD}(m, t)$ in order to show that the dimension and the entries of the basis depend on the parameters m, t. To illustrate the lattice construction, we give below the matrix $B_{BD}(2, 1)$ where the coefficient vectors form the rows of the basis.

$$B_{BD}(2,1) = \left(\begin{array}{c|c|cc|ccc|ccc}
 & 1 & x & xy & x^2 & x^2y & x^2y^2 & y & xy^2 & x^2y^3 \\
\hline
e^2 & e^2 & & & & & & & & \\
\hline
xe^2 & & e^2X & & & & & & & \\
fe & -e & eAX & eXY & & & & & & \\
\hline
x^2e^2 & & & & e^2X^2 & & & & & \\
xfe & & -eX & & eAX^2 & eX^2Y & & & & \\
f^2 & 1 & -2AX & -2XY & A^2X^2 & 2AX^2Y & X^2Y^2 & & & \\
\hline\hline
ye^2 & & & & & & & e^2Y & & \\
yfe & & & eAXY & & & & -eY & eXY^2 & \\
yf^2 & & & -2AXY & & A^2X^2Y & 2AX^2Y^2 & Y & -2XY^2 & X^2Y^3
\end{array}\right) \qquad (5.3)$$

Running LLL algorithm we can obtain two short vectors b_1, b_2 which by inequality 3.1 satisfy

$$\|b_1\|, \|b_2\| \leq 2^{\frac{w}{2}} det(\mathcal{L}_{BD})^{\frac{1}{w-1}}$$

where w is the dimension of the lattice constructed.Thus a sufficient condition in order to apply lemma 5.3.3 is

$$2^{\frac{w}{2}} det(\mathcal{L}_{BD})^{\frac{1}{w-1}} \leq \frac{e^m}{\sqrt{w}}.$$

Some tedious computations show that the determinant and the dimension of the lattice $\mathcal{L}_{BD}$ have the following values respectively

$$\begin{aligned} det(\mathcal{L}_{BD}) &= e^{\frac{5+4\delta}{12}m^3 + \frac{3+2\delta}{4}tm^2 + \frac{mt^2}{4} + o(m^3)}, \\ w &= \frac{m^2}{2} + tm + o(m^2). \end{aligned}$$

Optimizing with respect to t and ignoring low degree terms gives the condition

$$-12\delta^2 + 28\delta - 7 < 0 \Rightarrow \delta < \frac{7}{6} - \frac{1}{3}\sqrt{7} \approx 0.284.$$

This means that if $\delta < 0.284$ or, equivalently $d < N^{0.284}$, one can find in time polynomial in $\log N$ the factorization of N and consequently break RSA.

Remark 5.3.4. A precise calculation of the determinant (including low degree terms) along with a detailed proof for the bound on δ can be found in [5, Appendix A]. As m (and consequently the dimension of the lattice) grows larger, δ converges to the value $\frac{7}{6} - \frac{\sqrt{7}}{3} \approx 0.2874$.

Remark 5.3.5. Constructing exactly the same lattice for an arbitrary α (and not necessarily 1), Boneh and Durfee [5, Section 6] prove that the condition for δ becomes $\delta < \frac{7}{6} - \frac{1}{3}\sqrt{1+6\alpha}$. This means that the above attack becomes even stronger if $\alpha < 1$. For instance if $e \approx N^{\frac{1}{2}}$ the attack can tolerate values of δ up to $\frac{1}{2}$.

Improving the bound to $d < N^{0.292}$

Boneh and Durfee further improve the bound for δ (for the case $\alpha \approx 1$) by considering a sublattice of $\mathcal{L}_{BD}$. They remove some of the rows of the corresponding basis matrix B_{BD} that contribute more to the volume of the lattice than others and look for small elements in the new sublattice.Computations become quite complicated since the removal of certain rows renders a non-square matrix, the determinant of whose cannot be trivially calculated. They introduce the new concept of *geometrically progressive matrices* and finally prove that the bound can be pushed to $\delta = 1 - \frac{1}{\sqrt{2}} \approx 0.292$. The details of their proof can be found in [5, Section 5].

The bound 0.292 is up to now the best known bound for cryptanalysis of low secret exponent RSA. However, the attack only works under the assumption that the polynomials returned by LLL are algebraically independent. This makes the method heuristic. In practice, nevertheless, no failure of the method is known for sufficiently large δ.

We just mention here that if one takes $t = 0$ (that is only x-shifts) one is supposed to get a bound $\delta = 0.25$ which reproduces Wiener's result. However, Blömer an May [2] observed that the method of Boneh and Durfee always fails when using only x-shifts.

Remark 5.3.6. In contrast to Wiener's attack which is completely inapplicable for $e > N^{\frac{3}{2}}$, the attack introduced by Boneh and Durfee becomes completely inefficient for values of e larger than $N^{1.875}$. To see that, recall that for arbitrary α the condition for δ is $\delta < \frac{7}{6} - \frac{1}{3}\sqrt{1+6\alpha}$. As α grows larger, the bound for δ approaches 0, value which is finally reached for $\alpha = \frac{15}{8}$.

Remark 5.3.7. The approach used assumed that p, q are balanced.In [5, Section 7], Boneh and Durfee also consider the case where $p < q$ and $p < N^{\beta}$ where $\beta \leq \frac{1}{2}$. Using a similar (heuristic) technique with the three unknows k, p, q and replacing every occurence of pq by N, they prove that the more unbalanced the primes p, q are, the more effective the low private exponent attacks become.

5.3.3 Blömer & May Low Private Exponent Attack (BM)

In 2001, Blömer and May [2] revisited the attack mounted by Boneh and Durfee.Their analysis yields a bound $\delta = 0.290$. While this bound is not better than $\delta = 0.292$ achieved by Boneh and Durfee, the approach presented by Blömer and May is significantly simpler.

They begin their analysis by choosing parameters m and t and then construct exactly the same lattice $\mathcal{L}_{BD}$ as Boneh and Durfee (before the removal of the rows) with corresponding basis $B_{BD}(m, t)$. Next they remove certain rows of B_{BD} to take an intermediate matrix $\bar{B}$. Let $\bar{\mathcal{L}}$ be the lattice spanned by $\bar{B}$.Unlike Boneh and Durfee, they go on by removing an equal number of columns in order to obtain a square matrix. Details on the construction can be found in [2, Section 4]. As an example, the following matrix corresponds to matrix 5.3 after the removal of certain rows and columns. We denote the final (eliminated) matrix constructed by Blömer and May as B_{BM} and the corresponding lattice $\mathcal{L}_{BM}$

$$B_{BM}(2,1) = \left(\begin{array}{c|cc|ccc|c}
 & x & xy & x^2 & x^2y & x^2y^2 & x^2y^3 \\
\hline
xe^2 & e^2X & & & & & \\
fe & eAX & eXY & & & & \\
\hline
x^2e^2 & & & e^2X^2 & & & \\
xfe & -eX & & eAX^2 & eX^2Y & & \\
f^2 & -2AX & -2XY & A^2X^2 & 2AX^2Y & X^2Y^2 & \\
\hline\hline
yf^2 & & -2AXY & & A^2X^2Y & 2AX^2Y^2 & X^2Y^3
\end{array}\right)$$

The row vectors of the the matrix B_{BM} are no longer the coefficient vectors of the polynomials $g_{i,k}(xX, yY)$ and $h_{j,k}(xX, yY)$ since we have also removed some columns from the initial basis matrix B_{BD} (notice that the basis constructed by Boneh and Durfee does not suffer from the same drawback since we have only removed rows and not columns).However in order to aplly lemma 5.3.3, it is necessary to ensure that the integer linear combination of bivariate polynomials evaluates to zero modulo e^m at the point (k, s).Blömer and May show how to associate the rows of B_{BM} matrix with the polynomials $g_{i,k}$ and $h_{j,k}$. This means that they show how to reconstruct a vector $\bar{u} \in \bar{\mathcal{L}}$ by a vector $u \in \mathcal{L}_{BM}$.

More significantly, they prove that short vectors $u \in \mathcal{L}_{BM}$ lead to short reconstruction vectors $\bar{u} \in \bar{\mathcal{L}}$. Expressed in a different way, the size of small vectors found in the eliminated lattice $\mathcal{L}_{BM}$ by LLL are the same size as those found in the original lattice $\bar{\mathcal{L}}$ up to a small correction term. A complete substantiation of the above claims can be found in [2, Section 4].

Remark 5.3.8. Although it yields a weaker bound than Boneh and Durfee method, the new approach followed by Blömer and May has some noteworthy advantages:

(a) It leads to simple proofs since one deals with square matrices which significantly simplifies detrminant calculations.

(b) It reduces the lattice dimension as a function of m and t which implies that one can get closer to the theoretical bound.

(c) It makes use of structural properties of the underlying polynomials which makes possible its extension to other lattice constructions using these polynomials.

5.3.4 Small CRT-Exponent Attacks

An alternative way in order to speed up RSA decryption/signature process without using a small private exponent d, is to use the Chinese Remander Theorem (CRT). Suppose that one chooses d such that both $d_p \equiv d \pmod{p-1}$ and $d_q \equiv d \pmod{\frac{q-1}{2}}$ are small. From now on we will call such an exponent d *small CRT-exponent* .One can then decrypt a ciphertext c fast if one computes $m_p \equiv c^{d_p} \pmod{p}$ and $m_q \equiv c^{d_q} \pmod{q}$ and then combine the results using CRT to obtain the unique $m \in \mathbb{Z}_N$ such that $m^d \equiv c \pmod{N}$. The attacks described in the previous subsections do not work in general since d is likely to be large even though d_p, d_q are small.For the general case (where p, q are arbitrary) there is no known polynomial time algorithm that breaks RSA if d_p, d_q are small.The best algorithm up to now is exponential in the bitsize of N ($O(\min(\sqrt{d_p}, \sqrt{d_q}))$).

In 2002, May [25] gave a polynomial time algorithm for a low CRT-Exponent Attack for the case where the primes p, q are unbalanced. In his paper, May presents two methods for attacking small CRT-exponent RSA. The first method is provable and works as long as $q < N^{\frac{3-\sqrt{5}}{2}} \approx N^{0.382}$ while the second one is heuritic but can tolerate larger values of d modulo $p-1$. Before presenting the main ideas of the aformentioned methods, we first formalize the CRT key generation procedure (algorithm 8) and the goal of the CRT-exponent attacks.

Algorithm 8: CRT Key Generation Process

```
Input: Public Key $(N, e)$ and plaintext $m$.
Output: Ciphertext $c$ corresponding to plaintext $m$.
begin
    Step 1: Fix the bitsize $n$ of the modulus $N$ and two positive
    parameters $\beta, \delta$ such that $\beta \leq \frac{1}{2}$ and $\delta \leq 1$.
    Step 2 (Modulus): Choose (randomly) primes $p, q$ with
    approximate bitsize $(1-\beta)n, \beta n$ respectively such that
    $\gcd(p-1, \frac{q-1}{2}) = 1$ and compute the modulus $N = pq$. Repeat step
    2 if $q \geq N^{\beta}$.
    Step 3 $(d_p, d_q)$: Choose $d_p \in \mathbb{Z}_{p-1}$ such that $d_p \leq N^{\delta}$ and
    $d_q \in \mathbb{Z}_{\frac{q-1}{2}}$ arbitrarily.
    Step 4 (CRT): Compute the unique $d$ mod $\frac{\phi(N)}{2}$ such that $d \equiv d_p$
    (mod $p-1$) and $d \equiv d_q$ (mod $\frac{q-1}{2}$).
    Step 5 (Public Exponent):Compute the inverse $e$ of $d$ in $\mathbb{Z}_{\frac{\phi(N)}{2}}$.
    Step 6 : Publish the pair $(N, e)$.
end
```

Remark 5.3.9. Notice that the requirement $\gcd(p-1, \frac{q-1}{2}) = 1$ in step 2 is necessary for the existence of the *unique* d mod $\frac{\phi(N)}{2}$ computed using the CRT in step 4.

The question answered by the two methods presented below is the following:"Up to which values of the parameters β and δ can one find the factorization of N efficiently given the public key pair (N, e)?"

Remark 5.3.10. Notice that the decryption and signature generation processes can be performed efficiently if the parameters are chosen according to the CRT Key Generation Algorithm and β, δ are small. Indeed, c^{d_p} (mod p) and c^{d_q} (mod q) can easily be computed since d_p and q are small. We can then compute c^d (mod N) using the CRT.

First method

According to the CRT Key generation algorithm 8 we know that

$$ed_p \equiv 1\,(mod p-1).$$

Hence, there exists $k \in \mathbb{Z}$ such that

$$ed_p + k(p-1) = 1 \Rightarrow ed_p - (k+1) = -kp, \quad \text{over } \mathbb{Z}. \tag{5.4}$$

Consider now the polynomial $f_p(x,y) = ex - y$ which by construction (see equation 5.4) has a root $(x_0, y_0) = (d_p, k+1)$ modulo p. In order to apply Coppersmith's results, we have to bound the values of the root pair $(d_p, k+1)$. By assumption $d_p \leq N^\delta$. In addition,

$$|k+1| = \left|\frac{ed_p - p}{p-1}\right| < \frac{ed_p}{p-1} < \frac{q-1}{2} d_p < N^{\beta+\delta}$$

where the last but one inequality holds since $e < \frac{\phi(N)}{2}$.If we let $X = N^\delta$ and $Y = N^{\beta+\delta}$, we have ended up with the bivariate modular equation $f_p \equiv 0 \, (mod p)$ for which we are searcing small solutions (x_0, y_0) such that $|x_0| \leq X$ and $|y_0| \leq Y$.In order to transform the modular equation to an equation over the integers, May uses lemma 5.3.3. In order to find a polynomial $f(x,y)$ with a sufficiently short corresponding coefficient vector, he first uses a two-dimensional lattice whose rows are the coefficient vectors of the polynomials $f_0(xX, yY)$ and $f_p(xX, yY)$ where $f_0(x,y) = Nx$.For each root pair (x_0, y_0) of f_p, every linear integer combination of this two polynomials obviously yields a polynomial f such that $f(x_0, y_0) \equiv 0$ mod p. The basis of the lattice is the following

$$B_p = \begin{bmatrix} NX & \\ eX & -Y \end{bmatrix}.$$

May first proves the following two lemmas.

Lemma 5.3.11. Let $X = N^\delta$ and $Y = N^{\beta+\delta}$ with

$$3\beta + 2\delta \leq 1 - \log_N 4.$$

Then the lattice $\mathcal{L}_p$ generated by B_p has a smallest vector u such that $\|u\| \leq \frac{p}{\sqrt{2}}$.

Proof. The condition of the theorem is equivalent to the following

$$3\beta + 2\delta \leq 1 - \log_N 4 \Rightarrow 1 + \beta + 2\delta \leq 2 - 2\beta - \log_N 4 \Rightarrow N^{1+\beta+2\delta} \leq \frac{N^{2-2\beta}}{4}.$$

By Minkowski's theorem (theorem 3.1.2), $\mathcal{L}_p$ contains a vector u such that

$$\|u\| < \sqrt{2}\sqrt{det(\mathcal{L}_p)} = \sqrt{2}\sqrt{NXY} = \sqrt{2N^{1+2\delta+\beta}} \leq$$
$$\leq \sqrt{2\frac{N^{2-2\beta}}{4}} < \sqrt{\frac{p^2}{2}} = \frac{p}{\sqrt{2}}$$

where in the last inequality we have used that $p > N^{1-\beta}$ since $q < N^\beta$. $\square$

Notice that since the lattice is two-dimensional, a lattice reduction algorithm in fact returns such a short vector u and does not just approximate it. This means that we can find a vector u such that $\|u\| < \frac{p}{\sqrt{2}}$ in polynomial time as long as $3\beta + 2\delta \leq 1 - \log_N 4$. May proved that such a vector u can directly lead to the factorization of N.

Lemma 5.3.12. Let $u = (c_0, c_1)B_p$ be a shortest vector in $\mathcal{L}_p$ with $\|u\| < \frac{p}{\sqrt{2}}$. Then $(|c_0|, |c_1|) = (|k|, qd_p)$.

Proof. $u = c_0(NX, 0) + c_1(eX, -Y)$ is the coefficient vector of the polynomial $f(xX, yY)$ where

$$f(x, y) = c_0Nx + c_1(ex - y).$$

Since by construction $f(xX, yY) < \frac{p}{\sqrt{2}}$ and $|d_p| < N^{\delta}, |k+1| < N^{\beta+\delta}$, we know that $(x_0, y_0) = (d_p, k+1)$ is a root over the integers (by lemma 5.3.3 with $\omega = 2$). This along with equation 5.4 gives

$$c_0Nd_p = -c_1(ed_p - (k+1)) = c_1kp \Rightarrow c_0qd_p = c_1k.$$

Assume that q does not divide k.This means that $\gcd(qd_p, k) = \gcd(d_p, k) = 1$ where the last equality is obvious by 5.4.

By equality $c_0qd_p = c_1k$, k divides c_0qd_p but is coprime with qd_p which gives $c_0 = ak$ for an integer a. Thus $c_1 = aqd_p$. But since u is the smallest vector in $\mathcal{L}_p$ it follows that $|a| = 1$ which completes the proof. □

May then uses lemmas 5.3.11 and 5.3.12 to prove the following theorem.

Theorem 5.3.13
Let (N, e) be the RSA public key pair with $N = pq$ and d the secret exponent. If $q < N^{\beta}, d_p \leq N^{\delta}$ and

$$3\beta + 2\delta \leq 1 - \log_N 4,$$

then N can be factored in time $O(\log^2 N)$.

Proof. Construct the lattice basis B_p and run Gauss reduction algorithm to find $u = (c_0, c_1) \cdot B_p$. Compute $\gcd(N, c_1) = q$. The total running time for Gauss reduction and computation of the greatest common divisor is $O(\log^2 N)$. □

Remark 5.3.14. In lemma 5.3.12 we assumed that q does not divide k. May proves that if q divides k then the results of the analysis are even stronger. This implies that the small CRT-exponent attack described above works for a larger range of values β, δ. In particular, May proves the following theorem. The proof can be found in [25, Theorem 6].

Theorem 5.3.15
Let (N, e) be the RSA public key pair with $N = pq$ and d the secret exponent. If $q < N^{\beta}, d_p \leq N^{\delta}$ and

$$k = qr \quad , \quad \beta + 2\delta \leq 1 - \log_N 4,$$

then N can be factored in time $O(\log^2 N)$.

The inequality in theorem 5.3.13 implies that the above method works only if $\beta < \frac{1}{3}$. May [25] extends this bound to the value $\beta < \frac{3-\sqrt{5}}{2} \approx 0.382$ by enriching the lattice that produces the small coefficient vector. In particular, May considers the following x-shift polynomials

$$g_{m,i,j}(x, y) = N^{max(0,m-j)} x^i f_p^j(x, y)$$

for the same f_p as in the previous approach. Now the root pair (x_0, y_0) of $f_p \equiv 0 \, mod \, p$ is the root of every linear integer combination of polynomials $g_{m,i,j}$ modulo p^m and not only modulo p. Let $\mathcal{L}_p'$ denote the new lattice produced by the basis B_p' whose rows are the coefficient vectors of the polynomials $g_{m,i,j}(xX, yY)$.

In order to construct the lattice, one have to choose two integers n (the dimension of the lattice) and m which is a function of n to be optimized. In order to to transform the modular equation to an equation over the integers, May applies again lemma 5.3.3 with $\omega = n$. Following argumentation similar to the previous analysis, May invokes LLL algorithm to prove the following lemma. The details of the proof can be found in [25, lemma 7].

Lemma 5.3.16. For every fixed $\epsilon > 0$, there are parameters n, N_0 such that for every $N \geq N_0$ the following holds: Let $X = \frac{n+1}{2} N^{\delta}$ and $Y = \frac{n+1}{2} N^{\beta+\delta}$ with

$$3\beta - \beta^2 + 2\delta \leq 1 - \epsilon.$$

Then using the LLL algorithm, we can find a vector $u \in \mathcal{L}_p'$ with norm smaller than $\frac{p^m}{\sqrt{n}}$, where m is a function of n.

We have thus ended up with a single bivariate polynomial $f(x, y)$. In order to recover the roots (x_0, y_0), May proves the following lemma (details to be found in [25, lemma 8]).

Lemma 5.3.17. Let $X = \frac{n+1}{2} N^{\delta}$, $Y = \frac{n+1}{2} N^{\beta+\delta}$ and $f_p(x, y) = ex - y$ be a polynomial with root (x_0, y_0) modulo p that satisfies $|x_0| \leq N^{\delta}, |y_0| \leq N^{\beta+\delta}$. Let u be a vector in $\mathcal{L}_p'(n)$ such that $\|u\| \leq \frac{p^m}{\sqrt{n}}$, where u is the coefficient vector of a polynomial $f(xX, yY)$. Then the polynomial $p(x, y) = y_0 x - x_0 y \in \mathbb{Z}[x, y]$ must divide $f(x, y)$. We can then factor f over $\mathbb{Z}[x, y]$ and consequently find p.

Combining lemmas 5.3.16 and 5.3.17 we get the following result:

Theorem 5.3.18
Let (N, e) be the RSA public key pair with $N = pq$ and d the secret exponent. If $q < N^{\beta}, d_p \leq N^{\delta}$ and

$$3\beta - \beta^2 + 2\delta \leq 1 - \epsilon,$$

where $\epsilon > 0$ arbitrary small for N suitably large, then N can be factored in time polynomial in $\log N$.

Remark 5.3.19. It is important to note that both approaches of the first method are provable in that they do not rely on algebraic independence of two polynomials returned by LLL. Indeed, notice that the roots are recovered using a single polynomial which means that the method entails no resultant computations.

Second method

In his second method, May uses two polynomials with short coefficient vectors in order to recover (x_0, y_0). Thus, in contrast to the first one, the second method is heuristic since it is based on Coppersmith's technique for finding small solutions to bivariate modular polynomial equations.

May begins by rearranging the equation $ed_p + k(p-1) = 1$ to get

$$(k+1)(p-1) - p = -ed_p \Rightarrow (k+1)(N-q) - N = -eqd_p.$$

This leads to the problem of finding small roots to the bivariate modular polynomial

$$f_e(y, z) = y(N - z) - N$$

with a known root $(y_0, z_0) = (k+1, q)$ modulo e. Define the bounds $Y = N^{\beta+\delta}$ and $Z = N^{\beta}$ so that $|y_0| \leq Y, |z_0| \leq Z$. For the construction of the corresponding lattice, May fixes a positive integer parameter m and considers the y- and z-shifted polynomials

$$g_{i,j}(y, z) = e^{m-i} y^j f_e^i(y, z) \, and$$
$$h_{i,j}(y, z) = e^{m-i} z^j f_e^i(y, z)$$

which all have the common root (x_0, y_0) modulo e^m. In order to apply lemma 5.3.3, May looks for two short vectors in the lattice spanned by the coefficient vectors of $g_{i,j}(yY, zZ)$ and $h_{i,j}(yY, zZ)$ for certain values of i and j. The condition for β, δ under which condition 2 of lemma 5.3.3 is satisfied, is given in the following lemma.

Lemma 5.3.20. For every fixed constant $\epsilon > 0$, there exist m, N_0 such that for every $N \geq N_0$ the following holds: Let $Y = N^{\beta+\delta}$ and $Z = N^{\beta}$ with

$$\frac{2}{3}(\beta + \sqrt{3\beta + \beta^2}) + \delta \leq 1 - \epsilon, \tag{5.5}$$

where ϵ is arbitrary small for N suitably large.Then using the LLL algorithm, we can find two vectors $u_1, u_2 \in \mathcal{L}$ such that $\|u_1\|, \|u_2\| \leq \frac{e^m}{\sqrt{dim(\mathcal{L})}}$.

This means that if condition 5.5 holds, we can find two polynomials $h_1(y,z)$ and $h_2(y,z)$ which have the root $(y_0,z_0) = (k+1,q)$ over the integers. Using resultant computations we can recover all roots z_0 and therefore q which yields the factorization of N. Of course, the method's success relies on the assumption that the resultant $res_y(h_1(y,z),h_2(y,z))$ is a non zero polynomial, which makes the method clearly heuristic.

Recent Small CRT-exponent Attacks

In 2006 , Bleichenbacher and May [1] extended May's second method and managed to improve the bound to $q < N^{0.468}$ (recall that the first two methods described previously yielded bounds $q < N^{0.382}$ and $q < N^{0.375}$ respectively).Their extension uses again Coppersmith's technique for finding small solutions to multivariate modular polynomial equations and is therefore heuristic.

Bleichenbacher and May use again the polynomial $f(x,y) = x(N-y)+N$ as in the second method (to be more precise, in his second method,May used the polynomial $f(x,y) = x(N-y)-N$ which is equivalent to $f(x,y) = x(N-y)+N$ if we replace $k+1$ with $-(k+1)$).In order to improve the bounds, they make additional use of the fact that the root (x_0,q) mod e they are searching for, contains the prime factor q. Consequently, they introduce a new variable z for the prime factor p and an additional equation $yz = N$.

The general approach is the same as in the second method above, that is, Bleichenbacher and May construct a lattice $\mathcal{L}$ with basis B and use LLL to get two sufficiently short vectors u_1,u_2 such that $\|u_1\|,\|u_2\| \leq \frac{e^m}{\sqrt{dim(\mathcal{L})}}$ in order to be able to apply lemma 5.3.3. Since we have introduced a third variable z,these two vectors are the coefficient vectors of two trivariate polynomials $f_1(xX,yY,zZ)$ and $f_2(xX,yY,zZ)$ which by lemma 5.3.3 both have the root (x_0,q,p) over the integers. Once these two trivariate polynomials are obtained, we can eliminate z by setting $z = \frac{N}{y}$ and then multiplying both f_1,f_2 with a suitable power of y to get two integer polynomials $\bar{f}_1,\bar{f}_2$ in the two variables x,y. We can then recover the small root pairs (x_0,y_0) (and consequently (x_0,q)) using resultant computations. This method will finally give q provided that the resultant $res_x(\bar{f}_1(x,y),\bar{f}_2(x,y))$ is not the zero polynomial.This assumption makes the method heuristic but still very powerful in practice.

The challenge is to construct the lattice in such a way that the short vectors returned by LLL are short enough to yield improved bounds for X,Y,Z.Having LLL in mind, this goal reduces to constructing a lattice with a determinant as small as possible.To that end, Bleichenbacher and May incorporate the following two ideas:

(a) They first multiply the polynomial $f(x,y) = x(N-y)+N$ by the monomial z^s for some parameter s that has to be optimized. This leads to the following collection of trivariate (instead of the initial bivariate $f(x,y)$) polynomials

$$g'_{i,j}(x,y,z) = z^s \cdot g_{i,j}(x,y) = e^{m-i}x^j z^s f_e^i(x,y)\, and$$
$$h'_{i,j}(x,y,z) = z^s \cdot h_{i,j}(x,y) = e^{m-i}y^j z^s f_e^i(x,y).$$

where for $g'_{i,j}(x,y,z)$, $i = 0,...,m; j = 0,...,m-i$ and for $h'_{i,j}(x,y,z)$, $i = 0,...,m; j = 1,...,t$. Notice that every monomial $x^i y^j$ $j \geq s$ in the initial collection of polynomials with coefficient $a_{i,j}$ is transformed into a monomial $x^i y^{j-s}$ with coefficient $a_{i,j}N^s$ in the new collection. Similarly if $j < s$ the monomial $x^i y^j$ is transformed into the monomial $x^i z^{s-j}$ with new coefficient $a_{i,j}N^j$. This implies that the coefficient vectors of $g'_{i,j}(xX,yY,zZ)$ and $h'_{i,j}(xX,yY,zZ)$ contain less powers of Y which decreases the determinant of the lattice spanned by these vectors. Nevertheless, the determinant is increased because now the coefficient vectors also include powers of Z since we have added the third variable z. The goal is to optimize this trade-off by choosing a value for the parameter s that will minimize the value of the determinant.

(b) The resulting lattice basis built from the coefficient vectors of polynomials $g'_{i,j}(xX,yY,zZ)$ and $h'_{i,j}(xX,yY,zZ)$ is lower triangular. This means that every polynomial in the new collection contributes to the determinant of the matrix with just one coefficient (the coefficient of the diagonal entry). If that coefficient has a factor N^j, we can eliminate this factor by multiplying the polynomial with the inverse of N^j mod e.Eliminating powers of N in the diagonal entries keeps the lattice determinant as small as possible.

The incorporation of these two ideas in the construction of the lattice,along with a suitable choice of the parameter s, yield the following sufficient condition for the bounds X, Y, Z. The proof can be found in [1, lemma 4, appendix A].

Lemma 5.3.21. Let $\epsilon > 0, t = \tau m$ and $s = \sigma m$.Let N and m be sufficiently large and

$$X^{2+3\tau}Y^{1+3(\tau-\sigma)(1+\tau-\sigma)}Z^{3\sigma^2} \leq e^{1+3\tau-\epsilon}. \tag{5.6}$$

Then on input B, LLL algorithm will output two vectors that are shorter than $\frac{e^m}{\sqrt{dim(\mathcal{L})}}$.

This means that under the condition 5.6, LLL returns in polynomial time, two vectors u_1, u_2 the corresponding polynomials of which satisfy condition 2 of lemma 5.3.3.

In order to obtain the final condition among β, δ and α where $d_p \leq N^\delta, q \leq N^\beta$ and $e = N^\alpha$, one has to express X, Y and Z in terms of β, δ and α and plug these values in the condition of lemma 5.3.21. Some straightforward but tedious computations lead to the following theorem.A detailed proof can be found in [1, theorem 5].We emphasize here that the following result is based on the assumption that the emerging resultants are nonzero polynomials and is therefore only conventionally characterized as theorem.

Theorem 5.3.22
Let $\epsilon > 0$ and N be sufficinetly large. Let $N = pq$ be an RSA-modulus with $q \leq N^{\beta}$ and $p \leq 2N^{1-\beta}$. Moreover, let $e = N^{\alpha}$ be an RSA-public exponent satisfying $ed_p \equiv 1 \,(mod p - 1)$ for some $d_p = N^{\delta}$ with

$$\delta \leq \frac{1}{3}(3 - 2\beta - \beta^2 - \sqrt{12\alpha\beta - 12\alpha\beta^2 + 4\beta^2 - 5\beta^3 + \beta^4}) - \epsilon.$$

Then N can be factored in polynomial time.

Comparison of the methods

Let us summarize the small CRT-exponent attacks presented above. Recall that the parameters δ, β are used to bound, in terms of N, the values d_p and q respectively ($d_p \leq N^{\delta}, q \leq N^{\beta}$).In table 5.1 below we present the various conditions for d_p under which N can be factored in polynomial time. In all methods q is bounded by N^{β}.

Method	**Reference**	**Condition**	**Comment**
1st Method	[25]	$d_p \leq N^{\frac{1-3\beta+\beta^2}{2}-\epsilon}$	Provable
2nd Method	[25]	$d_p \leq N^{1-\frac{2}{3}(\beta+\sqrt{3\beta+\beta^2})-\epsilon}$	Heuristic
Recent Results	[1]	$d_p \leq N^{\frac{1}{3}(3-2\beta-\beta^2-\sqrt{12\alpha\beta-12\alpha\beta^2+4\beta^2-5\beta^3+\beta^4})-\epsilon}$	Heuristic

Table 5.1: Review of small CRT-exponent attacks

5.3.5 Small Private Exponent Attacks to RSA Schemes with Moduli $N = p^r q$

In 2004, May [28] presented two low private exponent attacks against RSA-like schemes with modulus $N = p^r q$. We will again assume that p, q are of the same bitsize as we did in section 5.2 where we presented factoring attacks against such schemes. May in these two attacks makes direct use of the theorems 5.2.2 and 5.2.8 presented in section 5.2.

First Attack

May's first attack is based on theorem 5.2.8 and can be summarized in the following result:

Theorem 5.3.23
Let $N = p^r q$, where $r \geq 2$ is a known constant and p, q are primes of the same

bitsize. Let $(e,d) \in \mathbb{Z} \times \mathbb{Z}^*_{\phi(N)}$ *be the public-key/secret-key pair satisfying* $ed \equiv 1\,(mod\,\phi(N))$*. Suppose that*

$$d \leq N^{\frac{r}{(r+1)^2}}.$$

Then N can be factored in polynomial time.

Proof. Since $N = p^r q$, $\phi(N) = p^{r-1}(p-1)(q-1)$. This means that there exists an integer k such that

$$ed - 1 = kp^{r-1}(p-1)(q-1). \tag{5.7}$$

Let E be the inverse of e modulo N (if such an inverse does not exist then $\gcd(E,N)$ is a non trivial factor of N and we can find p,q using the same arguments as in proof of theorem 5.2.8).Then $Ee = 1 + cN$ for some $c \in \mathbb{N}$. If we multiply 5.7 with E we get

$$\begin{aligned} &d(cN+1) - E = E \cdot kp^{r-1}(p-1)(q-1) \Rightarrow \\ &d - E = (Ekp^{r-2}(p-1)(q-1) - cp^{r-1}qd)p. \end{aligned}$$

This means that E is a multiple of p up to an additive error $d \leq N^{\frac{r}{(r+1)^2}}$. We define $t = Ekp^{r-2}(p-1)(q-1) - cp^{r-1}qd$ and we consider the next two possible cases:

1. $p^{r-1}q$ divides t. Then $p^{r-1}q$ divides $Ekp^{r-2}(p-1)(q-1)$ (the second term of t is obviously a multiple of $p^{r-1}q$). This means that pq divides $Ek(p-1)(q-1)$. Since by assumption $\gcd(E,N) = 1$ this means that pq divides $k(p-1)(q-1)$ that is $k(p-1)(q-1) = c'pq$. Then 5.7 becomes $ed - 1 = c'N$ which, in combination with $eE - 1 = cN$, gives that $E \equiv d\,(mod\,N)$. Since $E, d < N$ this implies that $d = E$ over the integers. But the knowledge of d suffices to factor N (there is a well known probabilistic polynomial time algorithm that computes p,q on input d and in the previous chapter we presented a detreministic polynomial time algorithm that performs the same computation).

2. $p^{r-1}q$ does not divide t. Then we can directly apply theorem 5.2.8 where t plays the role of k and recover the factorization of N.

□

Algorithm 9 summarizes the steps that yield the factorization of N.

Second Attack

For his second attack, May follows a slightly different approach using Coppersmith's Generalized theorem for univariate modular equations (theorem 5.2.2).His attack can be summarized in the following theorem.

Algorithm 9: May's First Attack for small d using a modulus $N = p^r q$

```
Input: $(N, e)$ where $N = p^r q$ and $ed \equiv 1 \,(mod\, \phi(N))$ for some
    $d \leq N^{\frac{r}{(r+1)^2}}$.
Output: The factors $p, q$ of $N$.
1. Compute $E = e^{-1}$ mod$N$. If the computation of $E$ fails, ouptut $p, q$ and EXIT.
2.Run algorithm of theorem 5.2.8 on input $E$. If the algorithm outputs $p, q$ EXIT.
3.Set $d = E$ and run a (probabilistic or deterministic) factorization algorithm on input $(N, e, d)$.
```

Theorem 5.3.24
Let $N = p^r q$, where $r \geq 2$ is a known constant and p, q are primes of the same bitsize. Let $(e, d) \in \mathbb{Z} \times \mathbb{Z}^*_{\phi(N)}$ be the public-key/secret-key pair satisfying $ed \equiv 1 \,(mod\, \phi(N))$. Suppose that

$$d \leq N^{(\frac{r-1}{r+1})^2}.$$

Then N can be factored in polynomial time.

Proof. We begin again with equation

$$ed - 1 = kp^{r-1}(p-1)(q-1), \quad \text{for some } k \in \mathbb{Z}.$$

Let $E = e^{-1} \,(mod\, N)$. Then again $Ee = 1 + cN$ for some $c \in \mathbb{N}$. If we multiply the equation $ed - 1 = kp^{r-1}(p-1)(q-1)$, with E and rearrange its terms we will get

$$d - E = (Ek(p-1)(q-1) - cdpq)p^{r-1}.$$

Define the polynomial $f_{p^{r-1}} = x - E$ which has the root $x_0 = d$ modulo p^{r-1}. In addition, since p, q are of the same bitsize, we know that $p \geq \frac{1}{2}q$.This gives

$$p^{r+1} = p^r p \geq \frac{p^r q}{2} = \frac{1}{2}N \Rightarrow$$
$$p^{r-1} \geq (\frac{1}{2}N)^{\frac{r-1}{r+1}} \geq \frac{1}{2}N^{\frac{r-1}{r+1}}.$$

We would like to be able to recover $x_0 = d$ in polynomial time. We will apply theorem 5.2.2 to $f_{p^{r-1}} = x - E$ in order to show that d can in fact be computed in polynomial time.The degree of $f_{p^{r-1}}$ is $\delta = 1$. Let $\beta = \frac{r-1}{r+1} - \frac{1}{\log N}$. We know that p^{r-1} is a divisor of N such that $p^{r-1} \geq \frac{1}{2}N^{\frac{r-1}{r+1}} = N^\beta$. In order to apply theorem 5.2.2 and recover d it remains to show that $d \leq c_N N^{\frac{\beta^2}{\delta}}$ for a constant c_N. If we choose $c_N = 4$ then we have

$$4N^{\frac{\beta^2}{\delta}} = 4N^{(\frac{r-1}{r+1})^2 - \frac{2(r-1)}{(r+1)\log N} + \frac{1}{\log^2 N}} \geq 4N^{(\frac{r-1}{r+1})^2 - \frac{2}{\log N}} = N^{(\frac{r-1}{r+1})^2} \geq d.$$

Thus we can recover d in polynomial time in $\log N$ and then use a (deterministic or probabilistic) polynomial time algorithm to obtain the factorization of N.This completes the proof of the theorem. $\square$

Algorithm 10 gives a compact description of the attack.

Algorithm 10: May's Second Attack for small d using a modulus $N = p^r q$

```
Input: $(N, e)$ where $N = p^r q$ and $ed \equiv 1\,(mod\,\phi(N))$ for some
        $d \leq N^{(\frac{r-1}{r+1})^2}$.
Output: The factors $p, q$ of $N$.
1. Compute $E = e^{-1}$ mod$N$. If the computation of $E$ fails, ouptut $p, q$
and EXIT.
2.Apply the algorithm of theorem 5.2.2 on input
$N, f_{p^{r-1}} = x - E, \beta = \frac{r-1}{r+1} - \frac{1}{\log N}$ and $c_N = 4$.This gives the value of $d$.
3.Run a (probabilistic or deterministic) factorization algorithm on
input $(N, e, d)$.
```

The bounds in theorems 5.3.23 and 5.3.24 imply that the first attack is more efficient if $r = 2$ while the second one yields better results for all integer values $r \geq 3$.

Remark 5.3.25. It is important to note here that in contrast to Wiener and Boneh & Durfee attacks, which required $e < \phi(N)$ and were completely inefficient for $e > N^{1.5}$ and $e > N^{1.875}$ respectively, none of the attacks presented above can be counteracted by choosing a larger public exponent e. Intuitively, this difference stems from the fact that all previous attacks required the computation of k in equation $ed - 1 = k\phi(N)$. In order to find k, one had to bound it first and consequently bound e. On the contrary, May's attacks do not require the computation of k and thus both k and e can be arbitrarily large. Instead, they take advantage of the fact that $r \geq 2$ and thus $\phi(N)$ and N share some common divisors.

Remark 5.3.26. Another interesting feature of the new attacks is that they can trivially be extended to partial key exposure attacks for d with known most significant bits (MSBs).Actually, it makes no difference to the attacker whether the most significant bits of d are zero (which impies that d is small) or known. On the contrary, Wiener and Boneh & Durfee attacks do not work when the MSBs of d are non-zero but known.

5.4 Partial Key-Exposure Attacks

In the previous section we examined attacks where the private key d was small. In this section we study the case where the value of d is arbitrary but we know a fraction of its bits. The major question we will try to answer is how many bits of d does an eavesdropper need in order to reconstruct all of d and thus break RSA. All the attacks described below are equally strong to the attacks described in sections 5.2 and 5.3 in that once an attack succeeds, the RSA is fully broken and the legitimate sender have to choose a new tuple (N, e, d) in order to re-establish a secure communication.

To give a practical perspective of the attacks in question, consider a computer system in which an RSA private key is stored. An intruder may attempt to attack the system in a variety of ways in order to obtain the private key. In many scenarios, an attacker using a side-channel attack,[4] either succeeds to obtain the most significant bits (MSBs) or least significant bits (LSBs) of d in consecutive order. Once a certain fraction of the bits is revealed, the attacker can efficiently compute all of d.

5.4.1 Boneh-Durfee-Frankel Partial-Key Exposure Attacks (BDF)

In 1998, Boneh,Durfee and Frankel [6] presented some partial key-exposure attacks on RSA.Their results include attacks where either some of the MSBs or LSBs of d are known and determine the conditions under which the knowledge of a fraction of consecutive bits of d is sufficient to fully recover d. Below we present in brief their results along with the main underlying ideas. In the following analysis we assume that p, q are balanced. However, similar results hold if p, q are not of the same bitsize.

Low Public Exponent

The first attack by Boneh ,Durfee and Frankel requires the knowledge of a quarter of the LSBs of private key d and is efficient only if the public exponent e is polynomially bounded by the bitsize of N. The starting point for the analysis is again the equation

$$ed - k\phi(N) = ed - k(N - s + 1) = 1 \tag{5.8}$$

where $s = p + q$ and $k \in \mathbb{N}$. We know that $d < \phi(N)$ which implies that $k < e$. Since p, q are assumed of the same bitsize, we have

$$4 < \frac{\sqrt{N}}{2} < q < p < 2\sqrt{N}.$$

[4]attack that is based on information gained from the physical implementation of a cryptosystem, rather than theoretical weaknesses in the algorithms . Timing information, power consumption, electromagnetic emanations or even sound can provide an extra source of information which can be exploited to break the system.

In addition $s = p + q < 3\sqrt{N}$ and thus $\phi(N) = N - s + 1 > N - 3\sqrt{N} > \frac{N}{2}$. The main result of BDF for low public exponent partial key-exposure attacks can be summarized in the following theorem.

Theorem 5.4.1 (**LSBs**)
Let $N = pq$ be an n-bit RSA modulus. Let $1 \leq e, d \leq \phi(N)$ satisfy $ed \equiv 1\,(mod\,\phi(N))$. if $N \equiv 3(mod\,4)$ and $e \leq 2^{\frac{n}{4}-3}$, then there exists an algorithm that, given the $\frac{n}{4}$ least significant bits of d, computes all of d in time polynomial in n and $e\log e$.

Proof Sketch. The knowledge of the $\frac{n}{4}$ least significant bits of d implies that we know some d_0 such that $d_0 \equiv d\,(mod\,2^{\frac{n}{4}})$. Reducing 5.8 modulo $2^{\frac{n}{4}}$ yields

$$ed_0 \equiv 1 + k(N - s + 1)\,(mod\,2^{\frac{n}{4}}).$$

If we replace q by $\frac{N}{p}$ and set $x = p$, we get the following univariate (quadratic) modular equation

$$ed_0 \equiv 1 + k(N - x - \frac{N}{x} + 1)\,(mod\,2^{\frac{n}{4}}) \tag{5.9}$$

$$kx^2 + (ed_0 - k(N+1) - 1)x + kN \equiv 0\,(mod\,2^{\frac{n}{4}}) \tag{5.10}$$

which has root $x_0 = p$ modulo $2^{\frac{n}{4}}$. Notice that if we could recover all the solutions x_0 of equation 5.10, then one of them would satisfy $x_0 \equiv p \pmod{2^{\frac{n}{4}}}$. We could then use theorem 5.2.5 to fully recover p and consequently N.

It only remains to descibe an algorithm that efficiently (with complexity $O(n^3)$) finds solutions to 5.10 and, in addition, bound the total number of solutions as k ranges in $\{1, ..., e\}$. We exhaustively examine all positive integers $k \leq e$. BDF transform 5.10 into a form for which there is an efficient algorithm that computes its roots.The details can be found in [16, Section 3 & appendix A].

In addition, they show that for each k' in the range $\{1, ..., e\}$ if k' is of the form $k' = 2^{t'_k}m$ where m is odd, then the number of roots of 5.10 for the candidate k' is upper bounded by $2^{2+t'_k}$. This means that an odd candidate k' can contribute to the total number of solutions with at most 4, a $k' = 4l + 2$ with at most 8 etc. Thus, as k' ranges over $\{1, ..., e\}$, $\Theta(e\log_2 e)$ solutions will be tested before the correct value of k is found.

The running time of the above approach is bounded by the number of solutions ($\Theta(e\log_2 e)$), the running time of algorithm in theorem 5.2.5 (which is polynomial in n) and the running time of the algorithm that finds the roots of 5.10 ($O(n^3)$) and is therefore polynomial. $\square$

Remark 5.4.2. Notice that the running time of the attack is heavily based on the value of the exponent e. The attack is polynomial in n (bitsize of N) only if e is polynomially bounded by n. That's why the attack is considered efficient only in the case where an exhaustive search in the range $\{1, ..., e\}$ is computationally feasible.

Remark 5.4.3. The requirement $N \equiv 3(mod\, 4)$ should not be ignored. When $N \equiv 3(mod\, 4)$, the above attack provably terminates in polynomial time and achieves the full recovery of d when $\frac{n}{4}$ of its LSBs are known. Notice that the condition $N \equiv 3(mod\, 4)$ is equivalent to $p \not\equiv q(mod\, 4)$. Expressed in another way, $p - q = 2r$ where r is odd. In the general case where $p - q = 2^{\alpha} r$ (with r odd), in 2004, Steinfeld and Zheng [36] showed that the above attack is efficent only when the number of known LSBs is at least $\frac{n}{4} + \alpha$.

Medium Public Exponent

Boneh, Durfee and Frankel extended their approach to public exponents with values in the interval $[N^{\frac{1}{4}}, N^{\frac{1}{2}}]$ when some of the MSBs (instead of LSBs) of d are known. They begin again with the equation 5.8 where $s = p + q$. The unknown variables are three (d, k and s). Unlike the previous case, k cannot be recovered by exhaustive search since it is an arbitrary element in the set $\{1, ..., e\}$ whose size is exponential in the bitsize of N. However they show that given sufficiently many MSBs of d and provided that $e < \sqrt{N}$, one can efficiently compute k up to a constant additive error. They first prove the following lemmas.

Lemma 5.4.4. Suppose that we are given a d_0 such that :

(i) $|e(d - d_0)| < c_1 N$, and

(ii) $ed_0 < c_2 N^{\frac{3}{2}}$.

Then the unique k satisfying $ed - k\phi(N) = 1$ is an integer in the range $[\tilde{k} - \Delta, \tilde{k} + \Delta]$ where $\tilde{k} = \frac{ed_0 - 1}{N}$ and $\Delta = 8c_2 + 2c_1$.

Proof.

$$\begin{aligned}
|\tilde{k} - k| = |k - \tilde{k}| &= \left| \frac{ed - 1}{\phi(N)} - \frac{ed_0 - 1}{N} \right| = \left| \frac{ed - 1}{\phi(N)} + \frac{ed_0 - 1}{\phi(N)} - \frac{ed_0 - 1}{\phi(N)} - \frac{ed_0 - 1}{N} \right| \\
&= \left| (ed_0 - 1)(\frac{1}{\phi(N)} - \frac{1}{N}) + \frac{e(d - d_0)}{\phi(N)} \right| < c_2 N^{\frac{3}{2}} (\frac{N - \phi(N)}{\phi(N) N}) + c_1 \frac{N}{\phi(N)} \\
&< c_2 \frac{4N^2}{\phi(N) N} + 2c_1 < 8c_2 + 2c_1.
\end{aligned}$$

since $N - \phi(N) < 3\sqrt{N} < 4\sqrt{N}$ and $\phi(N) > \frac{N}{2}$. This means that k is an integer in the range $[\tilde{k} - \Delta, \tilde{k} + \Delta]$ and the proof is complete. □

Lemma 5.4.5. Let $N = pq$ be an n-bit RSA modulus and $1 \leq e, d \leq \phi(N)$ satisfy $ed \equiv 1\, (mod\, \phi(N))$. In addition, let t be an integer in the range $\{0, ..., \frac{n}{2}\}$. Suppose that $2^t < e < 2^{t+1}$ and that we know the t most significant bits of d. Then we can efficiently compute the unique k satisfying 5.8 up to a constant additive error.

Proof. Since we know the t most significant bits of d, we know an integer d_0 such that $|d - d_0| < 2^{n-t}$. This means that d_0 satisfies $|e(d - d_0)| < e2^{n-t} < 2^{t+1+n-t} = 2 \cdot 2^n < 2N$. In addition $d_0 < N$ which means that $ed_0 < 2N^{\frac{3}{2}}$. Thus we can apply lemma 5.4.4 with $c_1 = c_2 = 2$ and search for k in the interval $[\tilde{k} - \Delta, \tilde{k} + \Delta]$ where $\Delta = 20$. □

Lemma 5.4.5 implies that we can recover the exact value of k by an exhaustive search and thus reduce the number of unknown variables in 5.8. By taking the above equation modulo e, we can further remove d and solve to find $s' = s \pmod e$. The main result of Boneh,Durfee and Frankel for MSBs known can be summarized in the following theorem.

Theorem 5.4.6
Let $N = pq$ be an n-bit RSA modulus and $1 \leq e, d \leq \phi(N)$ satisfy $ed \equiv 1 \, (mod \, \phi(N))$.

(a) Suppose that e is a prime in the range $\{2^t, ..., 2^{t+1}\}$ with $\frac{n}{4} \leq t \leq \frac{n}{2}$.Given the t most significant bits of d we can factor N in time polynomial in n.

(b) In general, suppose that $e \in \{2^t, ..., 2^{t+1}\}$ is the product of at most r known distinct primes with $\frac{n}{4} \leq t \leq \frac{n}{2}$.Given the t most significant bits of d we can factor N in time polynomial in n and 2^r.

Proof Sketch. The assumptions of the theorem satisfy lemma 5.4.5 which means that we can search for k in constant size range. In order to get the factorization of N, for each candidate $k' \in \{\tilde{k} - \Delta, ..., \tilde{k} + \Delta\}$ we do the following:

1. Compute $s' \equiv N + 1 + k'^{-1} \, (mod \, e)$. Notice that if we consider equation 5.8 modulo e, for a candidate k' we get

$$\begin{aligned} ed - k'(N - s + 1) = 1 &\Rightarrow k'(N - s + 1) + 1 \equiv 0 \, (mod \, e) \\ &\Rightarrow s' \equiv s \equiv N + 1 + k'^{-1} \, (mod \, e). \end{aligned}$$

   In addition equation 5.8 implies that $\gcd(k, e) = 1$ which means that we can remove every k' such that $\gcd(k, e) \neq 1$ from the candidate list.

2. Compute a root p' mod e for x in the quadratic equation

$$x^2 - s'x + N \equiv 0 \, (mod \, e). \tag{5.11}$$

   Since $s' \equiv s \, (mod \, e)$ then one of the solutions p' of 5.11 will satisfy $p' \equiv p \mod e$.

3. Once we have found a p' such that $p' \equiv p \mod e$ with $e \geq 2^{\frac{n}{4}} = N^{\frac{1}{4}}$, we can apply theorem 5.2.5 to fully recover p.

The execution of steps 1 and 3 are quite straightforward. It only remains to show how one can extract the roots of equation 5.11 efficiently.

(a) If e is a prime there are well known (probabilistic) polynomial time algorithms that find the corresponding roots.

(b) If e is a composite with r distinct prime factors $p_1, ..., p_r$,then we can solve the corresponding quadratic equations for each prime factor p_i and combine the solutions with Chinese Remainder Theorem to get the overall solutions. Thus, the running time of the algorithm depends on the number of the solutions of the quadratic equation. Since e has r distinct prime factors, there are at most 2^r solutions to consider. That's why the running time of the attack is polynommial in n and 2^r.

□

Interestingly, Boneh, Durfee and Frankel [16] prove that the full recovery of d is possible even if the factorization of e is not known.However, their results are weaker than in the case where the factorization of e is known.In particular they prove the following theorem , the proof of which can be found in [16, Section 4.2].

Theorem 5.4.7
Let $N = pq$ be an n-bit RSA modulus and $1 \leq e, d \leq \phi(N)$ satisfy $ed \equiv 1\,(mod\,\phi(N))$. Let t be integer in the range $\{0, ..., \frac{n}{2}\}$. Suppose e is in the range $\{2^t, ..., 2^{t+1}\}$ and $k > \epsilon \cdot e$ for some $\epsilon > 0$. Then there is an algorithm with running time polynomial in $n, \frac{1}{\epsilon}$ that given the $n - t$ MSBs of d finds all of d.

5.4.2 Blömer-May Partial Key-Exposure Attacks

In 2003, Blömer and May [3] presented some new partial key exposure attacks on RSA. Their results improve the bounds proved by Boneh, Durfee and Frankel. Apart from attacks with known most or least significant bits of d, they also present attacks with known MSBs or LSBs of CRT-exponent d_p. Below, we summarize the main results of Blömer and May and outline the underlying ideas of their proofs.The analysis presented assumes that p, q are of the same bitsize.

MSBs Known

Unlike Boneh, Durfee and Frankel who used the method for finding p, q when some of the bits of p are known, Blömer and May make direct use of Coppersmith's method for finding small solutions to modular multivariate equations. Consequently, Blömer and May manage to improve the bounds by relaxing the requirement that k in equation 5.8 is known exactly. This requirement restricted the method's usability to public exponents e with $e \leq N^{\frac{1}{2}}$.

Their main result for known MSBs is given in the following theorem. We note again here that since the method is heuristic, the theorem's validity depends on the non-occurence of zero-polynomials throughout the resultant computations.

Theorem 5.4.8
For every $\epsilon > 0$ there exists an integer N_0 such that for every $N > N_0$ the following holds:
Let (N, e) be an RSA public key, where $e = N^{\alpha}$ is in the range $[N^{\frac{1}{2}}, N^{\frac{\sqrt{6}-1}{2}}]$. Given an approximation $\tilde{d}$ of d with

$$|d - \tilde{d}| \leq N^{\frac{1}{8}(5-2\alpha-\sqrt{36\alpha^2+12\alpha-15})-\epsilon},$$

one can factor N in time polynomial in $\log N$.

Proof Sketch. Blömer and May do not try to fully determine k in the equation $ed - k\phi(N) = 1$. Instead, they use $\tilde{k} = \frac{e\tilde{d}-1}{N+1}$ as an approximation of k. Let $\delta = \frac{1}{8}(5 - 2\alpha - \sqrt{36\alpha^2 + 12\alpha - 15}) - \epsilon$. Then

$$\begin{aligned}|k - \tilde{k}| &= \left|\frac{ed-1}{\phi(N)} - \frac{e\tilde{d}-1}{N+1}\right| \\ &= \left|\frac{(ed-1)(N+1) - (e\tilde{d}-1)(N+1-(p+q))}{\phi(N)(N+1)}\right| \\ &\leq \left|\frac{e(d-\tilde{d})}{\phi(N)}\right| + \left|\frac{(p+q)(e\tilde{d}-1)}{\phi(N)(N+1)}\right| \leq \frac{e}{\phi(N)}(N^{\delta} + 3N^{-\frac{1}{2}}\tilde{d})\end{aligned}$$

where we have used the fact that $p + q < 3\sqrt{N}$ since p, q are balanced. Thus $\frac{(p+q)(e\tilde{d}-1)}{N+1} \leq 3N^{-\frac{1}{2}}e\tilde{d}$.
We consider the following two cases:

1. The term N^{δ} dominates $N^{-\frac{1}{2}}\tilde{d}$. Then

$$|k - \tilde{k}| \leq \frac{e}{\phi(N)}(N^{\delta} + 3N^{-\frac{1}{2}\tilde{d}}) \leq \frac{4N^{\delta}e}{\phi(N)} \leq 8N^{\delta+\alpha-1}$$

since $\phi(N) \geq \frac{N}{2}$. But the conditions $\delta = \frac{1}{8}(5 - 2\alpha - \sqrt{36\alpha^2 + 12\alpha - 15}) - \epsilon$ and $\alpha \geq \frac{1}{2}$ give that $\delta + a - 1 \leq 0$. That is we can easily determine k from $\tilde{k}$. If we determine k then we can compute $p + q = N + 1 + k^{-1} \bmod e$. On the other hand, $e \geq N^{\frac{1}{2}}$ and therefore (since $p + q \leq 3N^{\frac{1}{2}}$) we can find $p + q$ over the integers and not only modulo e.

2. The term $N^{-\frac{1}{2}}\tilde{d}$ dominates N^{δ}. Then

$$|k - \tilde{k}| \leq \frac{4eN^{-\frac{1}{2}}\tilde{d}}{\phi(N)} = \frac{4N^{\alpha-\frac{1}{2}}\tilde{d}}{\phi(N)} \leq 4N^{\alpha-\frac{1}{2}}.$$

Let $d_0 = d - \tilde{d}$ and $k_0 = k - \tilde{k}$. Then reformulating 5.8 we get

$$e(\tilde{d} + d_0) - 1 = (\tilde{k} + k_0)\phi(N)$$

or equivalently

$$ed_0 + (\tilde{k} + k_0)(p + q - 1) + e\tilde{d} - 1 = (\tilde{k} + k_0)N. \qquad (5.12)$$

The above equation motivates the definition of the following trivariate polynomial.

$$f_N(x, y, z) = ex + (\tilde{k} + y)z + e\tilde{d} - 1$$

with a root $(x_0, y_0, z_0) = (d_0, k_0, p + q - 1)$ modulo N. If we define $X = N^\delta, Y = 4N^{\alpha - \frac{1}{2}}$ and $Z = 3N^{\frac{1}{2}}$ then we have that $|x_0| \leq X, |y_0| \leq Y$ and $|z_0| \leq Z$.
Following the typical approach for multivariate modular equations, Blömer and May define the following polynomials

$$g_{i,j,k}(x, y, z) = x^{j-k} z^k N^i f_N^{m-i}(x, y, z) \quad for \begin{cases} 0 \leq i \leq m \\ 0 \leq j \leq i, and \\ 0 \leq k \leq j \end{cases}$$

$$h_{i,j,k}(x, y, z) = x^j y^k N^i f_N^{m-i}(x, y, z) \quad for \begin{cases} 0 \leq i \leq m \\ 0 \leq j \leq i, and \\ 1 \leq k \leq t \end{cases}$$

for a value t to be optimized.Notice that all these polynomials contain all of the roots of f_N modulo N^m. Next, they construct a lattice $\mathcal{L}$ the rows of which correspond to the coefficient vectors of the polynomials $g_{i,j,k}(xX, yY, zZ)$ and $h_{i,j,k}(xX, yY, zZ)$. In order to derive the condition for X, Y and Z, they apply Howgrave-Graham's lemma for the trivariate case (natural extension of lemma 5.3.3 to three variables). The computation of the determinant and the optimization with respect to t gives the following lemma (the proof can be found in [3, Lemma 8, p.11]).

Lemma 5.4.9. Let $X = N^\delta, Y = N^{\alpha - \frac{1}{2}}$ and $Z = N^{\frac{1}{2}}$. Then, using the LLL algorithm, one can find three linearly independent vectors in $\mathcal{L}$ with norm smaller than $\frac{N^m}{\sqrt{dim(\mathcal{L})}}$ provided that $\delta \leq \frac{1}{8}(5 - 2\alpha - \sqrt{36\alpha^2 + 12\alpha - 15})$.

The above lemma implies that the condition of the theorem ensures that we can find in polynomial time three polynomials $h_1(x, y, z)$, $h_2(x, y, z)$ and $h_3(x, y, z)$ with the common root (x_0, y_0, z_0). In order to recover the factorization of N, we just have to find $z_0 = p + q - 1$. To that end, we take the resultants $g_1 = res_x(h_1, h_2), g_2 = res_x(h_1, h_3)$ which are bivariate polynomials in y, z. In order to eliminate y as well we take the resultant $g' = res_y(g_1, g_2)$. If none of the g', g_1, g_2 is the zero polynomial, then we can extract the root z_0 and consequently factor N.

This completes the proof. □

LSBs Known

Blömer and May also presented two methods for known LSBs. In their first method, they present an attack that works for all but a negligible fraction of the public exponent $e < N^{\frac{1}{2}}$. Their approach makes use of the linear independence of two sufficiently short vectors in the lattice and does not use Coppersmith's heuristic technique. This, interestingly, leads to a rigorous method. In particular, they prove the following theorem.

Theorem 5.4.10
Let N be an RSA modulus and let $0 < \alpha, \epsilon < \frac{1}{2}$. For all but a $O(\frac{1}{N^{\epsilon}})$-fraction of the public exponent e in the interval $[3, N^{\alpha}]$ the following holds: Let d be the private exponent. Given d_0, M satisfying $d = d_0$ mod M with

$$N^{\alpha+\frac{1}{2}+\epsilon} \leq M \leq 2N^{\alpha+\frac{1}{2}+\epsilon}.$$

Then N can be factored in polynomial time.

Proof Sketch. Let $d = d_1 M + d_0$ where d_1 is unknown. If we replace d in equation 5.8 and reorder the terms we get the following equation

$$ed_1 M + k(p+q-1) - 1 + ed_0 = kN. \tag{5.13}$$

which in turn motivates the definition of the polynomial

$$f_N(x, y) = eMx + y + ed_0$$

with a root $(x_0, y_0) = (d_1, k(p+q-1)-1)$ modulo N. In addition

$$k = \frac{ed-1}{\phi(N)} < \frac{ed}{\phi(N)} < e \leq N^{\alpha}.$$

This implies that $k(p+q-1) - 1 < N^{\alpha} \cdot 3N^{\frac{1}{2}} = 3N^{\frac{1}{2}+\alpha}$.Moreover,

$$d_1 = \frac{d-d_0}{M} < \frac{N}{M} \leq \frac{N}{N^{\frac{1}{2}+\alpha+\epsilon}} = N^{\frac{1}{2}-\alpha-\epsilon}.$$

Thus, we can set the bounds $X = N^{\frac{1}{2}-\alpha-\epsilon}$ and $Y = 3N^{\frac{1}{2}+\alpha}$. In order to recover (x_0, y_0), Blömer and May again use Howgrave-Graham's lemma for the bivariate case (lemma 5.3.3) and transform the modular equation to an equation over the integers. For that, they use the auxiliary polynomials N and Nx and construct the following 3-dimensional lattice $\mathcal{L}$ with basis

$$B = \begin{bmatrix} N & & \\ 0 & NX & \\ ed_0 & eMX & Y \end{bmatrix}.$$

Again, the goal is to find two small linearly independent vectors $(a_0, a_1, a_2)B$ and $(b_0, b_1, b_2)B$ both having norm smaller than $\frac{N}{\sqrt{3}}$. Since the lattice is three dimensional, we can in fact compute two shortest linearly independent vectors in polynomial time. It only remains to show that $\mathcal{L}$ contains indeed two such vectors. To that end, they prove (the proof can be found in [3, lemma 10,p.14]) the following lemma.

Lemma 5.4.11. Given N, α, ϵ as defined in theorem 5.4.10. Then for all but $O(N^{\alpha-\epsilon})$ choices of e in the interval $[3, N^{\alpha}]$ the following holds: Let $X = N^{\frac{1}{2}-\alpha-\epsilon}$ and $Y = 3N^{\frac{1}{2}+\alpha}$. Then the lattice $\mathcal{L}$ contains two linearly independent vectors with norm less than $\frac{N}{\sqrt{3}}$.

Once we find these two vectors $\vec{a} = (a_0, a_1, a_2)$ and $\vec{b} = (b_0, b_1, b_2)$, we have the following equations over the integers

$$a_0N + a_1Nx_0 + a_2f_N(x_0, y_0) = 0,$$
$$b_0N + b_1Nx_0 + b_2f_N(x_0, y_0) = 0.$$

Blömer and May go on to solve the above system without using resultant computations. Since $f_N(x_0, y_0) = kN$, the above equations can be written as

$$\begin{aligned} a_1x_0 + a_2k &= -a_0, \\ b_1x_0 + b_2k &= -b_0. \end{aligned} \tag{5.14}$$

An important observation is that the linear independence of $\vec{a}, \vec{b}$ along with 5.14 ,implies the linear independence of $(a_1, a_2), (b_1, b_2)$ (suppose in contrast that $(a_1, a_2) = \lambda(b_1, b_2)$ for some $\lambda \in \mathbb{R}$, then equations 5.14 would give that $a_0 = \lambda b_0$ which contradicts the hypothesis that $\vec{a}, \vec{b}$ are lineraly independent). Thus, we can determine (x_0, k) as the unique solution of the linear system 5.14.Then we can compute y_0 by the relation $y_0 = kN - eMx_0 - ed_0$ and finally get the factorization of N from the relation $p + q - 1 = \frac{y_0+1}{k}$. This completes the proof.Note that the above result is rigorous since it does not use resultant computations. □

In their second method, they generalize the 3-dimensional approach to multidimensional lattices. They manage to improve the bound for the public exponent up to $e < N^{\frac{7}{8}}$. However, unlike their first method which is provable, this one is based on Coppersmith's multivariate approach and is therefore heuristic. Their main result can be summarized in the following theorem

Theorem 5.4.12
For every $\epsilon > 0$ there exists N_0 such that for every $N \geq N_0$ the following holds: Let (N, e) be an RSA public key with $e = N^{\alpha} \leq N^{\frac{7}{8}}$. Let d be the private exponent. Then given d_0, M satisfying $d \equiv d_0 \, (mod\, M)$ with

$$M \geq N^{\frac{1}{6}+\frac{1}{3}\sqrt{1+6\alpha}+\epsilon},$$

one can factor N in polynomial time.

Proof Sketch. Starting again with equation 5.8 and plugging in the value $d = d_1M + d_0$ we get

$$k(N-(p+q-1)) - ed_0 + 1 = eMd_1. \tag{5.15}$$

This motivates the definition of the polynomial

$$f_{eM}(y,z) = y(N-z) - ed_0 + 1$$

with the root $(y_0, z_0) = (k, p+q-1)$ modulo eM. As in the previous theorem, we set the bounds $Y = N^\alpha$ and $Z = 3N^{\frac{1}{2}}$ such that $|x_0| \leq X, |y_0| \leq Y$. Next, Blömer and May define the polynomials

$$g_{i,j}(y,z) = y^j(eM)^i f_{eM}^{m-i}(y,z) \quad for \begin{cases} 0 \leq i \leq m, \; and \\ 0 \leq j \leq i, \end{cases}$$
$$h_{i,j}(y,z) = z^j(eM)^i f_{eM}^{m-i}(y,z) \quad for \begin{cases} 0 \leq i \leq m, \; and \\ 1 \leq j \leq t \end{cases}$$

for some integers m and t (to be optimized later). They go on by constructing a lattice $\mathcal{L}(M)$ with basis $B(m)$ consisting of the coefficient vectors of the polynomials $g_{i,j}(yY, zZ)$ and $h_{i,j}(yY, zZ)$. Obviously, all the integer linear combinations of $g_{i,j}$ and $h_{i,j}$ have the root (y_0, z_0) modulo $(eM)^m$ which means that the first condition of lemma 5.3.3 is satisfied. For the satisfaction of the second condition, Blömer and May [3, lemma 12, p. 16] prove the following lemma.

Lemma 5.4.13. Let e, M be as defined in theorem 5.4.12. Suppose that $Y = N^\alpha$ and $Z = 3N^{\frac{1}{2}}$. Then LLL algorithm finds at least two vectors in $\mathcal{L}(M)$ with norm smaller than $\frac{(eM)^m}{\sqrt{dim(\mathcal{L}(M))}}$.

Thus the condition of theorem 5.4.12 guarantees that one can find in polynomial time two polynomials $f_1(y,z)$ and $f_2(y,z)$ with a root $(y_0, z_0) = (k, p+q-1)$ over the integers and consequently recover the factorization of N using resultant computations. Of course, the attack is heuristic. □

Known MSBs/LSBs and CRT-Exponents

Blömer and May extended their approach to fast RSA variants where the values $d_p = d\,(mod\,p-1)$ and $d_q = d\,(mod\,q-1)$ are used in decryption process instead of d. The attacks presented make use of a result due to Howgrave-Graham according to which an approximation of kp for some unknown k with error bound $N^{\frac{1}{4}}$ suffices to factor N. The attacks are provable since they do not rely on the assumption of non zero resultants. Their results for both LSBs and MSBs known are presented in the following theorem.

Theorem 5.4.14
Let (N, e) be an RSA pair with $N = pq, e = N^\alpha$ and secret exponent d. Let, in addition, $d_p = d\,(mod\,p-1)$.

(a) *Given* d_0, M *such that* $e = N^{\alpha} \in [1, poly(\log N)]$, $d_0 = d_p \,(mod\, M)$ *and* $M \geq N^{\frac{1}{4}}$, *or*

(b) *Given* $\tilde{d_p}$ *such that* $\alpha \in [0, \frac{1}{4}]$ *and* $|d_p - \tilde{d_p}| < N^{\frac{1}{4}-\alpha}$,

then N *can be factored in time polynomial in* $\log N$.

Proof. By definition of the RSA variant, we know that $ed_p - 1 = k(p-1)$ for some $k \in \mathbb{N}$. In addition,since $d_p < p - 1$, $k = \frac{ed_p - 1}{p-1} < e = N^{\alpha}$.

(a) If we write $d_p = d_1 M + d_0$ then we have $d_1 < \frac{d_p}{M} < \frac{p}{M} \leq 2N^{\frac{1}{4}}$. The above equation can then be rewritten as follows:

$$ed_0 + k - 1 = kp - eMd_1.$$

If $\gcd(eM, N) \neq 1$ then we obtain the factorization of N directly. Otherwise let E be the inverse of eM modulo N. Then $E \cdot eM = 1 + cN$ for some $c \in \mathbb{N}$. Multiplying the above equation with E we get

$$E(ed_0 + k - 1) = (Ek - cqd_1)p - d_1.$$

Suppose we know the value of k. Then we have an approximation of a multiple of p up to an additive error d_1 with $d_1 < 2N^{\frac{1}{4}}$. In addition q divides $Ek - cqd_1$ if and only if q divides Ek. Since $\gcd(E, N) = \gcd(eM, N) = 1$ this condition reduces to the condition q divides k. But if $e < q$ then q cannot divide k (recall that $k < e$). Thus we can apply theorem 5.2.3 and recover p as soon as we have found the right k. It only remains to show how to find k. This can be achieved by a brute force search in the interval $[1, e)$. Then for each value of k we run the algorithm of theorem 5.2.3 and for the right value we get the factorization of N. By hypothesis the size of e is polynomially bounded by $\log N$ which makes the whole attack work in time polynomial in $\log N$. This completes the first part of the proof.

(b) Since p, q are balanced and $k < N^{\alpha}$ where $\alpha \leq \frac{1}{4}$, q cannot divide k. The equation $ed_p - 1 = k(p-1)$ gives $kp = ed_p - 1 + k$. Define $\tilde{p} = e\tilde{d} - 1$. Then

$$\begin{aligned} |\tilde{p} - kp| &= |e\tilde{d} - 1 - (ed_p - 1 + k)| = |e(\tilde{d} - d_p) - k| \\ &\leq e|\tilde{d} - d_p| + |k| \leq N^{\alpha} \cdot N^{\frac{1}{4}-\alpha} + N^{\alpha} \leq 2N^{\frac{1}{4}}. \end{aligned}$$

We can then apply theorem 5.2.3 to factor N.

This completes the proof. □

5.4.3 RSA-like Schemes with moduli $N = p^r q$

In 2004, May [28] presented some partial key-exposure attacks to RSA Schemes with moduli $N = p^r q$. These attacks stem directly from the respective low private exponent attacks presented in subsection 5.3.5 . May shows that in such schemes it makes no difference whether the MSBs of d are zero or known to the attacker. Consequently, unlike the partial key-exposure attacks against the original RSA scheme, these attacks work for public exponents e of arbitrary size.

MSBs Known

The attacks for MSBs known are directly derived from theorems 5.3.23 and 5.3.24. We summarize May's results in the following theorem.

Theorem 5.4.15
Let $N = p^r q$, where $r \geq 2$ is a known constant and p, q are primes of the same bitsize. Let $(e, d) \in \mathbb{Z} \times \mathbb{Z}^*_{\phi(N)}$ be the public-key/secret-key pair satisfying $ed \equiv 1 \,(mod\, \phi(N))$. Given $\tilde{d}$ such that

$$|d - \tilde{d}| \leq N^{\frac{r}{(r+1)^2}}, \quad or \quad |d - \tilde{d}| \leq N^{(\frac{r-1}{r+1})^2},$$

one can factor N in (probabilistic) polynomial time.

Proof. For both conditions we begin with equation $ed - 1 = k\phi(N)$.

1. ($|d - \tilde{d}| \leq N^{\frac{r}{(r+1)^2}}$). Then we know that

$$e(d - \tilde{d}) + e\tilde{d} - 1 = kp^{r-1}(p-1)(q-1)$$

for some $k \in \mathbb{N}$. Multiplying the above equation with $E = e^{-1}$ modulo N ($eE = 1 + cN$ for some $c \in \mathbb{N}$)we get

$$(d - \tilde{d}) + E(e\tilde{d} - 1) = (Ekp^{r-2}(p-1)(q-1) - cp^{r-1}q(d - \tilde{d}))p \qquad (5.16)$$

which means that $E(e\tilde{d}-1)$ is a multiple of p up to an additive error $|(d-\tilde{d})| \leq N^{\frac{r}{(r+1)^2}}$. Using completely similar argumentation as in the proof of theorem 5.3.23 May proves that $E(e\tilde{d} - 1)$ yields the factorization of N.

2. ($|d - \tilde{d}| \leq N^{(\frac{r-1}{r+1})^2}$). Then if we rewrite 5.16 in a slightly different way, we get

$$(d - \tilde{d}) + E(e\tilde{d} - 1) = (Ek(p-1)(q-1) - cpq(d - \tilde{d}))p^{r-1}.$$

May, then defines the polynomial $f_{p^{r-1}}(x) = x + E(e\tilde{d} - 1)$ which has a small root $x_0 = d - \tilde{d}, |x_0| < N^{(\frac{r-1}{r+1})^2}$ modulo p^{r-1}. The rest of the proof is identical to the proof of theorem 5.3.24 where the polynomial in question is $f_{p^{r-1}}(x) = x + E(e\tilde{d} - 1)$ instead of $f_{p^{r-1}}(x) = x - E$.

This completes the proof. □

LSBs Known

In a similar way, May proves the following results for known LSBs of the private exponent d when the modulus is of the form $N = p^r q$.

Theorem 5.4.16
Let $N = p^r q$, where $r \geq 2$ is a known constant and p, q are primes of the same bitsize. Let $(e, d) \in \mathbb{Z} \times \mathbb{Z}^*_{\phi(N)}$ be the public-key/secret-key pair satisfying $ed \equiv 1\,(mod\,\phi(N))$. Given d_0, M with $d = d_0$ mod M and

$$M \geq N^{1-\frac{r}{(r+1)^2}}, \quad or \quad M \geq N^{\frac{4r}{(r+1)^2}}$$

one can factor N in (probabilistic) polynomial time.

Proof. We start by writing d as $d = d_1 M + d_0$.

1. $(M \geq N^{1-\frac{r}{(r+1)^2}})$.In that case $d_1 = \frac{d-d_0}{M} < \frac{N}{M} \leq N^{\frac{r}{(r+1)^2}}$. The equation $ed - 1 = k\phi(N)$ can then be written as

$$ed_1 M + ed_0 - 1 = kp^{r-1}(p-1)(q-1), \quad \text{for some } k \in \mathbb{N}.$$

If we multiply the above equation with $E = (eM)^{-1}$ mod N ($eME = 1 + cN$ for some $c \in \mathbb{N}$) we get

$$d_1 + E(ed_0 - 1) = (Ekp^{r-2}(p-1)(q-1) - cp^{r-1}qd_1)p. \tag{5.17}$$

which means that $E(ed_0 - 1)$ is a multiple of p up to an additive error $|d_1| \leq N^{\frac{r}{(r+1)^2}}$. The rest of the proof follows the proof of theorem 5.3.23

2. $(M \geq N^{1-\frac{4r}{(r+1)^2}})$. Now $d_1 < \frac{N}{M} \leq N^{1-\frac{4r}{(r+1)^2}} = N^{(\frac{r-1}{r+1})^2}$ The equation 5.17 can be rewritten as

$$d_1 + E(ed_0 - 1) = (Ek(p-1)(q-1) - cpqd_1)p^{r-1}.$$

This motivates the definition of the polynomial $f_{p^{r-1}}(x) = x + E(ed_0 - 1)$ which has a small root $x_0 = d_1, |x_0| < N^{(\frac{r-1}{r+1})^2}$ modulo p^{r-1}. The rest of the proof goes on like the proof of theorem 5.3.24.

This completes the proof. □

LSBs/MSBs Known and CRT-Exponents

May also presents partial key-exposure attacks for RSA-like Schemes with moduli $N = p^r q$ when the values $d_p = d \pmod{p-1}$ and $d_q = d \pmod{q-1}$ are used in the decryption process instead of d itself.These attacks, which are in a way a generalization of the attacks by Blömer and May [3] (presented in 5.4.2), work efficiently for small public exponent e and for known both MSBs and LSBs. The following theorem summarizes the results of May's attack.

Theorem 5.4.17
Let $N = p^r q$, where $r \geq 1$ is a known constant and p, q are primes of the same bitsize. Let e be the public key and let d_p satisfy $ed_p \equiv 1$ (mod $p-1$).

1. *Given d_0 and M such that $d_0 \equiv d_p$ (mod M) with $M \geq 2N^{\frac{1}{(r+1)^2}}$ and provided that e is polynomially bounded by $\log N$, or*
2. *Given $\tilde{d}$ such that $|d_p - \tilde{d}| \leq N^{\frac{r}{(r+1)^2}-\alpha}$ and provided that $\alpha = \log_N(e) \in [0, \frac{r}{(r+1)^2}]$,*

then N can be factored in time polynomial in $\log N$.

Proof. The equation $ed_p \equiv 1 \pmod{p-1}$ implies that

$$ed_p - 1 = k(p-1) \text{ for some } k \in \mathbb{Z}.$$

In addition, since $d_p < p - 1$ the above equation gives that $k < e$. Finally, since p, q are assumed balanced, we know that $p \leq 2N^{\frac{1}{r+1}}$.

1. Let us write $d_p = d_1 M + d_0$ with $d_1 = \frac{d_p - d_0}{M} < \frac{p}{M} \leq \frac{2N^{\frac{1}{r+1}}}{2N^{\frac{1}{(r+1)^2}}} = N^{\frac{r}{(r+1)^2}}$. We rewrite the above equation as follows:

$$ed_1 M + ed_0 + k - 1 = kp.$$

   We compute $E = (eM)^{-1}$, that is $EeM = 1 + cN$ for some $c \in \mathbb{N}$. Thus the above equation becomes

$$d_1 + E(ed_0 + k - 1) = (Ek - cp^{r-1}qd_1)p.$$

   Since k is unknown, we first do a brute force search for k in the interval $[1, e)$. For each possible value of k we run the algorithm of theorem 5.2.8 to recover the factorization of N. Notice that the conditions of theorem 5.2.8 are satified since the additive error d_1 satisfies $|d_1| < N^{\frac{r}{(r+1)^2}}$ and $p^{r-1}q = \Omega(N^{\frac{r}{r+1}})$ cannot divide $k < e$ which is polynomially bounded by $\log N$. In addition the number of values k to be tested is polynomial in $\log N$ which makes the whole attack polynomial in $\log N$.

2. The equation $ed_p - 1 = k(p-1)$ gives $kp = ed_p + k - 1$. Thus

$$\begin{aligned}|err| =& |kp - e\tilde{d}| = |e(d_p - \tilde{d}) + k - 1| \leq |e(d_p - \tilde{d})| + |k-1| \\ &\leq N^{\frac{r}{(r+1)^2}-\alpha} \cdot N^\alpha + N^\alpha \leq 2N^{\frac{r}{(r+1)^2}}.\end{aligned}$$

   Thus $e\tilde{d}$ is a multiple of p up to an additive error $|err| \leq 2N^{\frac{r}{(r+1)^2}}$. In addition, $k < e < N^{\frac{r}{(r+1)^2}}$ which means that k cannot be a multiple of $p^{r-1}q = \Omega(N^{\frac{r}{r+1}})$. We can then apply theorem 5.2.8 to factor N.

This completes the proof. □

Index

Bibliography

[1] Daniel Bleichenbacher and Alexander May. "New Attacks on RSA with Small Secret CRT-Exponents". In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2006.

[2] Johannes Blömer and Alexander May. "Low Secret Exponent RSA Revisited". In Silverman [35], pages 4–19.

[3] Johannes Blömer and Alexander May. "New Partial Key Exposure Attacks on RSA". In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 27–43. Springer, 2003.

[4] Dan Boneh. "Twenty years of attacks on the RSA cryptosystem". *"Notices of the American Mathematical Society (AMS)"*, 46(2):203–213, 1999.

[5] Dan Boneh and Glenn Durfee. "Cryptanalysis of RSA with Private Key Less than $^{0.292}$". In *EUROCRYPT*, pages 1–11, 1999.

[6] Dan Boneh, Glenn Durfee, and Yair Frankel. "An Attack on RSA Given a Small Fraction of the Private Key Bits". In Kazuo Ohta and Dingyi Pei, editors, *ASIACRYPT*, volume 1514 of *Lecture Notes in Computer Science*, pages 25–34. Springer, 1998.

[7] Dan Boneh, Glenn Durfee, and Nick Howgrave-Graham. "Factoring $\mathrm{N} = \mathrm{p}^{\mathrm{r}}\mathrm{q}$ for Large r". In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 326–337. Springer, 1999.

[8] Matthew Cary. "Lattice Basis Reduction, Algorithms and Applications". February 2002.

[9] Don Coppersmith. "Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known". In *EUROCRYPT*, pages 178–189, 1996.

[10] Don Coppersmith. "Finding a Small Root of a Univariate Modular Equation". In *EUROCRYPT*, pages 155–165, 1996.

[11] Don Coppersmith. "Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities". *J. Cryptology*, 10(4):233–260, 1997.

[12] Don Coppersmith. "Finding Small Solutions to Small Degree Polynomials". In Silverman [35], pages 20–31.

[13] Don Coppersmith, Matthew K. Franklin, Jacques Patarin, and Michael K. Reiter. "Low-Exponent RSA with Related Messages". In *EUROCRYPT*, pages 1–9, 1996.

[14] Jean-Sébastien Coron. "Finding Small Roots of Bivariate Integer Polynomial Equations Revisited". In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 492–505. Springer, 2004.

[15] Jean-Sebastien Coron and Alexander May. "Deterministic Polynomial Time Equivalence of Computing the RSA Secret Key and Factoring". Cryptology ePrint Archive, Report 2004/208, 2004. `http://eprint.iacr.org/`.

[16] Yair Frankel Dan Boneh, Glenn Durfee. Exposing an rsa private key given a small fraction of its bits.

[17] Whitfield Diffie and Martin Hellman. "new directions in cryptography". *IEEE Transactions on Information Theory*, 22:644–654, 1976. URL: `http://cr.yp.to/bib/entries.html#1976/diffie`.

[18] Glenn Durfee and Phong Q. Nguyen. "Cryptanalysis of the RSA Schemes with Short Secret Exponent from Asiacrypt '99". In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 14–29. Springer, 2000.

[19] Cynthia Dwork. "Lattices and Their Applications to Cryptography". Lecture Notes, Stanford University, June 1998.

[20] Matthew K. Franklin and Michael K. Reiter. "A Linear Protocol Failure for RSA with Exponent Three". Presented in rump session, Crypto 95, but not in the proceedings.

[21] Johan Hastad. "Solving simultaneous modular equations of low degree". *SIAM Journal on Computing*, 17:336–341, 1988. URL: `http://www.nada.kth.se/ johanh/papers.html`.

[22] Jason Hinek. "Lattices Attacks in Cryptography: A Partial Overview". School of Computer Science, University of Waterloo, Canada, October 2004.

[23] Nick Howgrave-Graham. "Finding Small Roots of Univariate Modular Equations Revisited". In Michael Darnell, editor, *IMA Int. Conf.*, volume 1355 of *Lecture Notes in Computer Science*, pages 131–142. Springer, 1997.

[24] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. "Factoring polynomials with rational coefficients". 261:515–534, 1982.

[25] Alexander May. "Cryptanalysis of Unbalanced RSA with Small CRT-Exponent". In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 242–256. Springer, 2002.

[26] Alexander May. *"New RSA Vulnerabilities Using lattice Reduction Methods"*. PhD thesis, University of Paderborn, http://www.informatik.tu-darmstadt.de/KP/alex.html, October 2003.

[27] Alexander May. "Computing the RSA Secret Key Is Deterministic Polynomial Time Equivalent to Factoring". In *CRYPTO*, pages 213–219, 2004.

[28] Alexander May. "Secret Exponent Attacks on RSA-type Schemes with Moduli N= $\mathrm{p^r q}$". In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 218–230. Springer, 2004.

[29] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *"Handbook of Applied Cryptography"*. CRC Press, Inc., Boca Raton, FL, USA, 1996.

[30] Danielle Micciancio. "Lattices in Cryptography and Cryptanalysis". Lecture Series, University of California, San Diego, Fall 2001.

[31] Maurice Mignotte. "An inequality about factors of polynomials". In *Mathematics of Computation*, volume 28.

[32] Gary L. Miller. "Riemann's Hypothesis and tests for primality". In *STOC '75: Proceedings of seventh annual ACM symposium on Theory of computing*, pages 234–239, New York, NY, USA, 1975. ACM Press.

[33] Oded Regev. "Lattices in Computer Science". Lecture Series, Tel Aviv University, Fall 2004.

[34] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". *Commun. ACM*, 21(2):120–126, 1978.

[35] Joseph H. Silverman, editor. *Cryptography and Lattices, International Conference, CaLC 2001, Providence, RI, USA, March 29-30, 2001, Revised Papers*, volume 2146 of *Lecture Notes in Computer Science*. Springer, 2001.

[36] Ron Steinfeld and Yuliang Zheng. On the security of rsa with primes sharing least-significant bits. *Appl. Algebra Eng. Commun. Comput.*, 15(3-4):179–200, 2004.

[37] Douglas Stinson. *"Cryptography: Theory and Practice,Second Edition"*. CRC Press, Inc., Boca Raton, FL, USA, 2002.

[38] Michael J. Wiener. "Cryptanalysis of short RSA secret exponents". *IEEE Transactions on Information Theory*, 36(3):553–558, 1990.