Chapter 19

Coppersmith's Method and Related Applications

An important application of lattice basis reduction is finding small solutions to polynomial equations $F(x) \equiv 0 \pmod{M}$ of degree $d > 1$. The main purpose of this chapter is to present some results of Coppersmith [141] on this problem. We also discuss finding small roots of bivariate integer polynomials and some other applications of these ideas.

In general, finding solutions to modular equations is easy if we know the factorisation of the modulus, see Section 2.12. However, if the factorisation of the modulus M is not known then finding solutions can be hard. For example, if we can find a solution to $x^2 \equiv 1 \pmod{M}$ that is not $x = \pm 1$ then we can split M. Hence, we do not expect efficient algorithms for finding all solutions to modular equations in general.

Suppose then that the polynomial equation has a "small" solution. It is not so clear that finding the roots is necessarily a hard problem. The example $x^2 \equiv 1 \pmod{M}$ no longer gives any intuition since the two non-trivial roots both have absolute value at least $\sqrt{M}$. As we will explain in this chapter, if $F(x) \equiv 0 \pmod{M}$ of degree d has a solution x_0 such that $|x_0| < M^{1/d-\epsilon}$ for small $\epsilon > 0$ then it can be found in polynomial-time. This result has a number of important consequences.

General references for the contents of this chapter are Coppersmith [141, 142], May [410, 411], Nguyen and Stern [463] and Nguyen [456].

19.1 Coppersmith's Method for Modular Univariate Polynomials

19.1.1 First Steps to Coppersmith's Method

We sketch the basic idea of the method, which goes back to Håstad. Let $F(x) = x^d + a_{d-1}x^{d-1} + \cdots + a_1 x + a_0$ be a monic polynomial of degree d with integer coefficients. Suppose we know that there exist one or more integers x_0 such that $F(x_0) \equiv 0 \pmod{M}$ and that $|x_0| < M^{1/d}$. The problem is to find all such roots.

Since $|x_0^i| < M$ for all $0 \leq i \leq d$ then, if the coefficients of $F(x)$ are small enough, one might have $F(x_0) = 0$ over $\mathbb{Z}$. The problem of finding integer roots of integer polynomials is easy: we can find roots over $\mathbb{R}$ using numerical analysis (e.g., Newton's method) and then round the approximations of the roots to the nearest integer and check whether they are solutions of $F(x)$.

The problem is therefore to deal with polynomials $F(x)$ having a small solution but whose coefficients are not small. Coppersmith's idea (in the formulation of Howgrave-Graham [296]) is to build from $F(x)$ a polynomial $G(x)$ that still has the same solution x_0, but which has coefficients small enough that the above logic does apply.

Example 19.1.1. Let $M = 17 \cdot 19 = 323$ and let

$$F(x) = x^2 + 33x + 215.$$

We want to find the small solution to $F(x) \equiv 0 \pmod{M}$ (in this case, $x_0 = 3$ is a solution, but note that $F(3) \neq 0$ over $\mathbb{Z}$).

We seek a related polynomial with small coefficients. For this example,

$$G(x) = 9F(x) - M(x+6) = 9x^2 - 26x - 3$$

satisfies $G(3) = 0$. This root can be found using Newton's method over $\mathbb{R}$ (or even the quadratic formula).

We introduce some notation for the rest of this section. Let $M, X \in \mathbb{N}$ and let $F(x) = \sum_{i=0}^{d} a_i x^i \in \mathbb{Z}[x]$. Suppose $x_0 \in \mathbb{Z}$ is a solution to $F(x) \equiv 0 \pmod{M}$ such that $|x_0| < X$. We associate with the polynomial $F(x)$ the row vector

$$b_F = (a_0, a_1 X, a_2 X^2, \cdots, a_d X^d). \tag{19.1}$$

Vice versa, any such row vector corresponds to a polynomial. Throughout this section we will interpret polynomials as row vectors, and row vectors as polynomials, in this way.

Theorem 19.1.2. *(Howgrave-Graham [296]) Let $F(x), X, M, b_F$ be as above (i.e., there is some x_0 such that $|x_0| \leq X$ and $F(x_0) \equiv 0 \pmod{M}$). If $\|b_F\| < M/\sqrt{d+1}$ then $F(x_0) = 0$.*

Proof: Recall the Cauchy-Schwarz inequality $(\sum_{i=1}^{n} x_i y_i)^2 \leq (\sum_{i=1}^{n} x_i^2)(\sum_{i=1}^{n} y_i^2)$ for $x_i, y_i \in \mathbb{R}$. Taking $x_i \geq 0$ and $y_i = 1$ for $1 \leq i \leq n$ one has

$$\sum_{i=1}^{n} x_i \leq \sqrt{n \sum_{i=1}^{n} x_i^2}.$$

Now

$$|F(x_0)| = \left| \sum_{i=0}^{d} a_i x_0^i \right| \leq \sum_{i=0}^{d} |a_i||x_0|^i \leq \sum_{i=0}^{d} |a_i| X^i \leq \sqrt{d+1}\|b_F\| < \sqrt{d+1} M/\sqrt{d+1} = M$$

where the third inequality is Cauchy-Schwarz. so $-M < F(x_0) < M$. But $F(x_0) \equiv 0 \pmod{M}$ and so $F(x_0) = 0$. □

Let $F(x) = \sum_{i=0}^{d} a_i x^i$ be a monic polynomial. We assume that $F(x)$ has at least one solution x_0 modulo M such that $|x_0| < X$ for some specified integer X. If $F(x)$ is not monic but $\gcd(a_d, M) = 1$ then one can multiply $F(x)$ by $a_d^{-1} \pmod{M}$ to make it monic. If $\gcd(a_d, M) > 1$ then one can split M and reduce the problem to two (typically easier) problems. As explained above, to find x_0 it will be sufficient to find a polynomial $G(x)$ with the same root x_0 modulo M but with sufficiently small coefficients.

To do this, consider the $d+1$ polynomials $G_i(x) = Mx^i$ for $0 \leq i < d$ and $F(x)$. They all have the solution $x = x_0$ modulo M. Define the lattice L with basis corresponding to these polynomials (by associating with a polynomial the row vector in equation (19.1)). Therefore, the basis matrix for the lattice L is

$$B = \begin{pmatrix} M & 0 & \cdots & 0 & 0 \\ 0 & MX & \cdots & 0 & 0 \\ \vdots & & & \vdots & \vdots \\ 0 & 0 & \cdots & MX^{d-1} & 0 \\ a_0 & a_1 X & \cdots & a_{d-1} X^{d-1} & X^d \end{pmatrix}. \tag{19.2}$$

Every element of this lattice is a row vector that can be interpreted as a polynomial $F(x)$ (via equation (19.1) such that $F(x_0) \equiv 0 \pmod{M}$.

Lemma 19.1.3. *The dimension of the lattice L defined in equation (19.2) above is $d+1$ and the determinant is*

$$\det(L) = M^d X^{d(d+1)/2}.$$

Exercise 19.1.4. Prove Lemma 19.1.3.

One now runs the LLL algorithm on this (row) lattice basis. Let $G(x)$ be the polynomial corresponding to the first vector $\underline{b}_1$ of the LLL-reduced basis (since every row of B has the form of equation (19.1) then so does $\underline{b}_1$).

Theorem 19.1.5. *Let the notation be as above and let $G(x)$ be the polynomial corresponding to the first vector in the LLL-reduced basis for L. Set $c_1(d) = 2^{-1/2}(d+1)^{-1/d}$. If $X < c_1(d) M^{2/d(d+1)}$ then any root x_0 of $F(x)$ modulo M such that $|x_0| \leq X$ satisfies $G(x_0) = 0$ in $\mathbb{Z}$.*

Proof: Recall that $\underline{b}_1$ satisfies

$$\|\underline{b}_1\| \leq 2^{(n-1)/4} \det(L)^{1/n} = 2^{d/4} M^{d/(d+1)} X^{d/2}.$$

For $\underline{b}_1$ to satisfy the conditions of Howgrave-Graham's theorem (i.e., $\|\underline{b}_1\| < M/\sqrt{d+1}$) it is sufficient that

$$2^{d/4} M^{d/(d+1)} X^{d/2} < M/\sqrt{d+1}.$$

This can be written as

$$\sqrt{d+1}\, 2^{d/4} X^{d/2} < M^{1/(d+1)},$$

which is equivalent to the condition in the statement of the Theorem. □

In other words, if $d = 2$ then it is sufficient that $X \approx M^{1/3}$ to find the small solution using the above method. If $d = 3$ then it is sufficient that $X \approx M^{1/6}$. This is the result of Håstad. Of course, LLL often works better than the worst-case bound, so small solutions x_0 may be found even when x_0 does not satisfy the condition of the Theorem.

Example 19.1.6. Let $M = 10001$ and consider the polynomial

$$F(x) = x^3 + 10x^2 + 5000x - 222.$$

One can check that $F(x)$ is irreducible, and that $F(x)$ has the small solution $x_0 = 4$ modulo M. Note that $|x_0| < M^{1/6}$ so one expects to be able to find x_0 using the above method. Suppose $X = 10$ is the given bound on the size of x_0. Consider the basis matrix

$$B = \begin{pmatrix} M & 0 & 0 & 0 \\ 0 & MX & 0 & 0 \\ 0 & 0 & MX^2 & 0 \\ -222 & 5000X & 10X^2 & X^3 \end{pmatrix}.$$

Running LLL on this matrix gives a reduced basis, the first row of which is

$$(444, 10, -2000, -2000).$$

The polynomial corresponding to this vector is

$$G(x) = 444 + x - 20x^2 - 2x^3.$$

Running Newton's root finding method on $G(x)$ gives the solution $x_0 = 4$.

19.1.2 The Full Coppersmith Method

The method in the previous section allows one to find small roots of modular polynomials, but it can be improved further. Looking at the proof of Theorem 19.1.5 one sees that the requirement for success is essentially $M^d X^{d(d+1)/2} = \det(L) < M^{d+1}$ (more precisely, it is $2^{d/4} M^{d/(d+1)} X^{d/2} < M/\sqrt{d+1}$). There are two strategies to extend the utility of the method (i.e., to allow bigger values for X). The first is to increase the dimension n by adding rows to L that contribute less than M to the determinant. The second is to increase the power of M on the right hand side. One can increase the dimension without increasing the power of M by using the so-called "x-shift" polynomials $xF(x), x^2F(x), \ldots, x^kF(x)$; Example 19.1.7 gives an example of this. One can increase the power of M on the right hand side by using powers of $F(x)$ (since if $F(x_0) \equiv 0 \pmod{M}$ then $F(x_0)^k \equiv 0 \pmod{M^k}$).

Example 19.1.7. Consider the problem of Example 19.1.6. The lattice has dimension 4 and determinant M^3X^3. The condition for LLL to output a sufficiently small vector is

$$2^{3/4} \left(M^3 X^6\right)^{1/4} \leq \frac{M}{\sqrt{4}}$$

which, taking $M = 10001$, leads to $X \approx 2.07$. (Note that the method worked for a larger value of x_0; this is because the bound used on LLL only applies in the worst case.)

Consider instead the basis matrix that also includes rows corresponding to the polynomials $xF(x)$ and $x^2F(x)$

$$B = \begin{pmatrix} M & 0 & 0 & 0 & 0 & 0 \\ 0 & MX & 0 & 0 & 0 & 0 \\ 0 & 0 & MX^2 & 0 & 0 & 0 \\ -222 & 5000X & 10X^2 & X^3 & 0 & 0 \\ 0 & -222X & 5000X^2 & 10X^3 & X^4 & 0 \\ 0 & 0 & -222X^2 & 5000X^3 & 10X^4 & X^5 \end{pmatrix}.$$

The dimension is 6 and the determinant is M^3X^{15}. The condition for LLL to output a sufficiently small vector is

$$2^{5/4}\left(M^3X^{15}\right)^{1/6} \leq \frac{M}{\sqrt{6}},$$

which leads to $X \approx 3.11$. This indicates that some benefit can be obtained by using x-shifts.

Exercise 19.1.8. Let $G(x)$ be a polynomial of degree d. Show that taking d x-shifts $G(x), xG(x), \ldots, x^{d-1}G(x)$ gives a method that works for $X \approx M^{1/(2d-1)}$.

Exercise 19.1.8 shows that when $d = 3$ we have improved the result from $X \approx M^{1/6}$ to $X \approx M^{1/5}$. Coppersmith [141] exploits both x-shifts and powers of $F(x)$. We now present the method in full generality.

Theorem 19.1.9. *(Coppersmith) Let $0 < \epsilon < \min\{0.18, 1/d\}$. Let $F(x)$ be a monic polynomial of degree d with one or more small roots x_0 modulo M such that $|x_0| < \frac{1}{2}M^{1/d-\epsilon}$. Then x_0 can be found in time bounded by a polynomial in $d, 1/\epsilon$ and $\log(M)$.*

Proof: Let $h > 1$ be an integer that depends on d and ϵ and will be determined in equation (19.3) below. Consider the lattice L corresponding (via the construction of the previous section) to the polynomials $G_{i,j}(x) = M^{h-1-j}F(x)^j x^i$ for $0 \leq i < d$, $0 \leq j < h$. Note that $G_{i,j}(x_0) \equiv 0 \pmod{M^{h-1}}$. The dimension of L is dh. One can represent L by a lower triangular basis matrix with diagonal entries $M^{h-1-j}X^{jd+i}$. Hence the determinant of L is

$$\det(L) = M^{(h-1)hd/2}X^{(dh-1)dh/2}.$$

Running LLL on this basis outputs an LLL-reduced basis with first vector $\underline{b}_1$ satisfying

$$\|\underline{b}_1\| < 2^{(dh-1)/4}\det(L)^{1/dh} = 2^{(dh-1)/4}M^{(h-1)/2}X^{(dh-1)/2}.$$

This vector corresponds to a polynomial $G(x)$ of degree $dh-1$ such that $G(x_0) \equiv 0 \pmod{M^{h-1}}$. If $\|\underline{b}_1\| < M^{h-1}/\sqrt{dh}$ then Howgrave-Graham's result applies and we have $G(x_0) = 0$ over $\mathbb{Z}$.

Hence, it is sufficient that

$$\sqrt{dh}2^{(dh-1)/4}M^{(h-1)/2}X^{(dh-1)/2} < M^{h-1}.$$

Rearranging gives

$$\sqrt{dh}2^{(dh-1)/4}X^{(dh-1)/2} < M^{(h-1)/2},$$

which is equivalent to

$$c(d,h)X < M^{(h-1)/(dh-1)}$$

where $c(d,h) = (\sqrt{dh}2^{(dh-1)/4})^{2/(dh-1)} = \sqrt{2}(dh)^{1/(dh-1)}$.

Now

$$\frac{h-1}{dh-1} = \frac{1}{d} - \frac{d-1}{d(dh-1)}.$$

Equating $(d-1)/(d(dh-1)) = \epsilon$ gives

$$h = ((d-1)/(d\epsilon)+1)/d \approx 1/(d\epsilon). \tag{19.3}$$

Note that $dh = 1 + (d-1)/(d\epsilon)$ and so $c(d,h) = \sqrt{2}(1+(d-1)/(d\epsilon))^{d\epsilon/(d-1)}$, which converges to $\sqrt{2}$ as $\epsilon \to 0$. Since $X < \frac{1}{2}M^{1/d-\epsilon}$ we require $\frac{1}{2} \leq \frac{1}{c(d,h)}$. Writing $x =$

$d\epsilon/(d-1)$ this is equivalent to $(1+1/x)^x \leq \sqrt{2}$, which holds for $0 \leq x \leq 0.18$. Therefore, assume $\epsilon \leq (d-1)/d$.

Rounding h up to the next integer gives a lattice such that if

$$|x_0| < \tfrac{1}{2} M^{1/d-\epsilon}$$

then the LLL algorithm and polynomial root finding leads to x_0.

Since the dimension of the lattice is $dh \approx 1/\epsilon$ and the coefficients of the polynomials $G_{i,j}$ are bounded by M^h it follows that the running time of LLL depends on $d, 1/\epsilon$ and $\log(M)$. □

Exercise 19.1.10. Show that the precise complexity of Coppersmith's method is $O((1/\epsilon)^9 \log(M)^3)$ bit operations (recall that $1/\epsilon > d$). Note that if one fixes d and ϵ and considers the problem as M tends to infinity then one has a polynomial-time algorithm in $\log(M)$.

We refer to Section 3 of [142] for some implementation tricks that improve the algorithm. For example, one can add basis vectors to the lattice corresponding to polynomials of the form $M^{h-1}x(x-1)\cdots(x-i+1)/i!$.

Example 19.1.11. Let $p = 2^{30}+3$, $q = 2^{32}+15$ and $M = pq$. Consider the polynomial

$$\begin{aligned} F(x) &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 \\ &= 1942528644709637042 + 1234567890123456789x + 9876543219876543 21x^2 + x^3, \end{aligned}$$

which has a root x_0 modulo M such that $|x_0| \leq 2^{14}$. Set $X = 2^{14}$. Note that $X \approx M^{1/4.4}$. One can verify that the basic method in Section 19.1.1 does not find the small root.

Consider the basis matrix (this is of smaller dimension than the lattice in the proof of Theorem 19.1.9 in the case $d = 3$ and $h = 3$)

$$\begin{pmatrix} M^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & M^2X & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & M^2X^2 & 0 & 0 & 0 & 0 \\ Ma_0 & Ma_1X & Ma_2X^2 & MX^3 & 0 & 0 & 0 \\ 0 & Ma_0X & Ma_1X^2 & Ma_2X^3 & MX^4 & 0 & 0 \\ 0 & 0 & Ma_0X^2 & Ma_1X^3 & Ma_2X^4 & MX^5 & 0 \\ a_0^2 & 2a_0a_1X & (a_1^2+2a_0a_2)X^2 & (2a_0+2a_1a_2)X^3 & (a_2^2+2a_1)X^4 & 2a_2X^5 & X^6 \end{pmatrix}.$$

The dimension is 7 and the determinant is M^9X^{21}. The first vector of the LLL reduced basis is

$$(-369928294330603367352173305173409792, 145105744202599483225996267040279 7568, \ldots)$$

This corresponds to the polynomial

$$\begin{aligned} &-369928294330603367352173305173409792 + 8856551770178191114867936220720 2x \\ &\quad -343998735725844172860857065 9x^2 + 4463580576455518968192 58x^3 \\ &\quad +4564259979987386926x^4 - 1728007960413053x^5 - 21177681998x^6 \end{aligned}$$

which has $x_0 = 16384 = 2^{14}$ as a real root.

Exercise 19.1.12. Let $M = (2^{20}+7)(2^{21}+17)$ and $F(x) = x^3 + (2^{25} - 2883584)x^2 + 46976195x + 227$. Use Coppersmith's algorithm to find an integer x_0 such that $|x_0| < 2^9$ and $F(x_0) \equiv 0 \pmod{M}$.

Remark 19.1.13. It is natural to wonder whether one can find roots right up to the limit $X = M^{1/d}$. Indeed, the $-\epsilon$ term can be eliminated by performing an exhaustive search over the top few bits of the root x_0. An alternative way to proceed is to set $\epsilon = 1/\log_2(M)$, break the range $|x_0| < M^{1/d}$ of size $2M^{1/d}$ into $M^{2\epsilon} = 4$ intervals of size $2M^{1/d-2\epsilon} = M^{1/d-\epsilon}$, and perform Coppersmith's algorithm for each subproblem in turn.

Another question is whether one can go beyond the boundary $X = M^{1/d}$. A first observation is that for $X > M^{1/d}$ one does not necessarily expect a constant number of solutions; see Exercise 19.1.14. Coppersmith [142] gives further arguments why $M^{1/d}$ is the best one can hope for.

Exercise 19.1.14. Let $M = p^2$ and consider $F(x) = x^2 + px$. Show that if $X = M^{1/2+\epsilon}$ where $0 < \epsilon < 1/2$ then the number of solutions $|x| < X$ to $F(x) \equiv 0 \pmod{M}$ is $2M^{\epsilon}$,

Exercise 19.1.15. Let $N = pq$ be a product of two primes of similar size and let $e \in \mathbb{N}$ be a small integer such that $\gcd(e, \varphi(N)) = 1$. Let $1 < a, y < N$ be such that there is an integer $0 \leq x < N^{1/e}$ satisfying $(a + x)^e \equiv y \pmod{N}$. Show that, given N, e, a, y one can compute x in polynomial-time.

19.2 Multivariate Modular Polynomial Equations

Suppose one is given $F(x, y) \in \mathbb{Z}[x, y]$ and integers X, Y and M and is asked to find one or more roots (x_0, y_0) to $F(x, y) \equiv 0 \pmod{M}$ such that $|x_0| < X$ and $|y_0| < Y$. One can proceed using similar ideas to the above, hoping to find two polynomials $F_1(x, y), F_2(x, y) \in \mathbb{Z}[x, y]$ such that $F_1(x_0, y_0) = F_2(x_0, y_0) = 0$ over $\mathbb{Z}$, and such that the resultant $R_x(F_1(x, y), F_2(x, y)) \neq 0$ (i.e., that $F_1(x, y)$ and $F_2(x, y)$ are **algebraically independent**). This yields a heuristic method in general, since it is hard to guarantee the independence of $F_1(x, y)$ and $F_2(x, y)$.

Theorem 19.2.1. *Let $F(x, y) \in \mathbb{Z}[x, y]$ be a polynomial of total degree d (i.e., every monomial $x^i y^j$ satisfies $i + j \leq d$). Let $X, Y, M \in \mathbb{N}$ be such that $XY < M^{1/d-\epsilon}$ for some $0 < \epsilon < 1/d$. Then one can compute (in time polynomial in $\log(M)$ and $1/\epsilon > d$) polynomials $F_1(x, y), F_2(x, y) \in \mathbb{Z}[x, y]$ such that, for all $(x_0, y_0) \in \mathbb{Z}^2$ with $|x_0| < X$, $|y_0| < Y$ and $F(x_0, y_0) \equiv 0 \pmod{M}$, one has $F_1(x_0, y_0) = F_2(x_0, y_0) = 0$ over $\mathbb{Z}$.*

Proof: We refer to Jutla [324] and Section 6.2 of Nguyen and Stern [463] for a sketch of the details. □

19.3 Bivariate Integer Polynomials

We now consider $F(x, y) \in \mathbb{Z}[x, y]$ and seek a root $(x_0, y_0) \in \mathbb{Z}^2$ such that both $|x_0|$ and $|y_0|$ are small. Coppersmith has proved the following important result.

Theorem 19.3.1. *Let $F(x, y) \in \mathbb{Z}[x, y]$ and let $d \in \mathbb{N}$ be such that $\deg_x(F(x, y)), \deg_y(F(x, y)) \leq d$. Write*

$$F(x, y) = \sum_{0 \leq i,j \leq d} F_{i,j} x^i y^j.$$

For $X, Y \in \mathbb{N}$ define

$$W = \max_{0 \leq i,j \leq d} |F_{i,j}| X^i Y^j.$$

If $XY < W^{2/(3d)}$ then there is an algorithm that takes as input $F(x, y), X, Y$, runs in time (bit operations) bounded by a polynomial in $\log(W)$ and 2^d, and outputs all pairs $(x_0, y_0) \in \mathbb{Z}^2$ such that $F(x_0, y_0) = 0$, $|x_0| \leq X$ and $|y_0| \leq Y$.

The condition in Theorem 19.3.1 is somewhat self-referential. If one starts with a polynomial $F(x,y)$ and bounds X and Y on the size of roots, then one can compute W and determine whether or not the algorithm will succeed in solving the problem.

Proof: (Outline) There are two proofs of this theorem, both of which are rather technical. The original by Coppersmith can be found in [141]. We sketch a simpler proof by Coron [150].

As usual we consider shifts of the polynomial $F(x,y)$. Choose $k \in \mathbb{N}$ (sufficiently large) and consider the k^2 polynomials

$$s_{a,b}(x,y) = x^a y^b F(x,y) \quad \text{for} \quad 0 \leq a,b < k$$

in the $(d+k)^2$ monomials $x^i y^j$ with $0 \leq i,j < d+k$. Coron chooses a certain set of k^2 monomials (specifically of the form $x^{i_0+i}y^{j_0+j}$ for $0 \leq i,j < k$ and fixed $0 \leq i_0, j_0 \leq d$) and obtains a $k^2 \times k^2$ matrix S with non-zero determinant M. (The most technical part of [150] is proving that this can always be done and bounding the size of M.)

One can now consider the $(d+k)^2$ polynomials Mx^iy^j for $0 \leq i,j < d+k$. Writing each polynomial as a row vector of coefficients, we now have a $k^2 + (d+k)^2$ by $(d+k)^2$ matrix. One can order the rows such that the matrix is of the form

$$\begin{pmatrix} S & * \\ MI_{k^2} & 0 \\ 0 & MI_w \end{pmatrix}$$

where $w = (d+k)^2 - k^2$, $*$ represents a $k^2 \times w$ matrix, and I_w denotes the $w \times w$ identity matrix.

Now, since $M = \det(S)$ there exists an integer matrix S' such that $S'S = MI_{k^2}$. Perform the row operations

$$\begin{pmatrix} I_{k^2} & 0 & 0 \\ -S' & I_{k^2} & 0 \\ 0 & 0 & I_w \end{pmatrix} \begin{pmatrix} S & * \\ MI_{k^2} & 0 \\ 0 & MI_w \end{pmatrix} = \begin{pmatrix} S & * \\ 0 & T \\ 0 & MI_w \end{pmatrix}$$

for some $k^2 \times w$ matrix T. Further row operations yield a matrix of the form

$$\begin{pmatrix} S & * \\ 0 & T' \\ 0 & 0 \end{pmatrix}$$

for some $w \times w$ integer matrix T'.

Coron considers a lattice L corresponding to T' (where the entries in a column corresponding to monomial x^iy^j are multiplied by X^iY^j as in equation (19.2)) and computes the determinant of this lattice. Lattice basis reduction yields a short vector that corresponds to a polynomial $G(x,y)$ with small coefficients such that every root of $F(x,y)$ is a root of $G(x,y)$ modulo M. If (x_0,y_0) is a sufficiently small solution to $F(x,y)$ then, using an analogue of Theorem 19.1.2, one infers that $G(x_0,y_0) = 0$ over $\mathbb{Z}$.

A crucial detail is that $G(x,y)$ has no common factor with $F(x,y)$. To show this suppose $G(x,y) = F(x,y)A(x,y)$ for some polynomial (we assume that $F(x,y)$ is irreducible, if not then apply the method to its factors in turn). Then $G(x,y) = \sum_{0 \leq i,j < k} A_{i,j} x^i y^j F(x,y)$ and so the vector of coefficients of $G(x,y)$ is a linear combination of the coefficient vectors of the k^2 polynomials $s_{a,b}(x,y)$ for $0 \leq a,b < k$. But this vector is also a linear combination of the rows of the matrix $(0\ T')$ in the original lattice. Considering the first k^2 columns (namely the columns of S), one has a linear dependence of the rows in S. Since $\det(S) \neq 0$ this is a contradiction.

It follows that the resultant $R_x(F,G)$ is a non-zero polynomial, and so one can find all solutions by finding the integer roots of $R_x(F,G)(y)$ and then solving for x.

To determine the complexity it is necessary to compute the determinant of T' and to bound M. Coron shows that the method works if $XY < W^{2/(3d)-1/k}2^{-9d}$. To get the stated running time for $XY < W^{2/(3d)}$ Coron proposes setting $k = \lfloor \log(W) \rfloor$ and performing exhaustive search on the $O(d)$ highest-order bits of x_0 (i.e., running the algorithm a polynomial in 2^d times). □

Example 19.3.2. Consider $F(x,y) = axy + bx + cy + d = 127xy - 1207x - 1461y + 21$ with $X = 30, Y = 20$. Let $M = 127^4$ (see below).

Consider the 13×9 matrix (this is taking $k = 2$ in the above proof and introducing the powers X^iY^j from the start)

$$B = \begin{pmatrix} aX^2Y^2 & bX^2Y & cXY^2 & dXY & 0 & 0 & 0 & 0 & 0 \\ 0 & aX^2Y & 0 & cXY & bX^2 & 0 & dX & 0 & 0 \\ 0 & 0 & aXY^2 & bXY & 0 & cY^2 & 0 & dY & 0 \\ 0 & 0 & 0 & aXY & 0 & 0 & bX & cY & d \\ MX^2Y^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & MX^2Y & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & & & & & & & & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & M \end{pmatrix}.$$

We take S to be the matrix

$$\begin{pmatrix} a & b & c & d \\ 0 & a & 0 & c \\ 0 & 0 & a & b \\ 0 & 0 & 0 & a \end{pmatrix}$$

corresponding to the monomials $x^{i_0+i}y^{j_0+j}$ for $0 \le i,j < 2$ and fixed $i_0 = j_0 = 1$. Note that $M = \det(S) = a^4 = 127^4$.

Rather than diagonalising using the method of the proof of Theorem 19.3.1 we compute the Hermite normal form of B. This gives the matrix

$$B' = \left(\begin{array}{cccc|ccccc} aX^2Y^2 & * & * & * & * & * & * & * & * \\ & aX^2Y & * & * & * & * & * & * & * \\ & & aXY^2 & * & * & * & * & * & * \\ & & & aXY & * & * & * & * & * \\ \hline & & & & 16129X^2 & 16129Y^2 & 100125X & 1064641Y & 202558777 \\ & & & & & 2048383Y^2 & * & * & * \\ & & & & & & 2048383X & * & * \\ & & & & & & & 260144641Y & * \\ & & & & & & & & 260144641 \\ \hline & & & & & & & & \end{array}\right)$$

where blanks are zeroes and $*$ denotes an entry whose value we do not bother to write down. Let L be the 5×5 diagonal matrix formed of columns 5 to 9 of rows 5 to 9 of B'. Performing LLL-reduction on L gives a matrix whose first row is

$$(-16129X^2, -16129Y^2, 1048258X, 983742Y, -28446222)$$

corresponding to the polynomial

$$G(x,y) = -16129x^2 - 16129y^2 + 1048258x + 983742y - 28446222.$$

Clearly $G(x,y)$ is not a multiple of $F(x,y)$, since it has no xy term. Computing resultants and factoring gives the solutions $(x,y) = (21,21)$ and $(23,19)$.

Exercise 19.3.3. The polynomial

$$F(x,y) = 131xy - 1400x + 20y - 1286$$

has an integer solution with $|x| < 30$ and $|y| < 20$. Use Coron's method as in Example 19.3.2 to find (x,y).

The results of this section can be improved by taking into account the specific shape of the polynomial $F(x,y)$. We refer to Blömer and May [72] for details.

Finally, we remark that results are also known for integer polynomials having three or more variables, but these are heuristic in the sense that the method produces a list of polynomials having small roots in common, but there is no guarantee that the polynomials are algebraically independent.

19.4 Some Applications of Coppersmith's method

19.4.1 Fixed Padding Schemes in RSA

As discussed in Chapter 1, it is necessary to use padding schemes for RSA encryption (for example, to increase the length of short messages and to prevent algebraic relationships between the messages and ciphertexts). One simple proposal for κ-bit RSA moduli is to take a κ' bit message and pad it by putting $(\kappa - \kappa' - 1)$ ones to the left hand side of it. This brings a short message to full length. This padding scheme is sometimes called **fixed pattern padding**; we discuss it further in Section 24.4.5.

Suppose short messages (for example, 128-bit AES keys K) are being encrypted using this padding scheme with $\kappa = 1024$. Then

$$\mathsf{m} = 2^{1024} - 2^{128} + K.$$

Suppose also that the encryption exponent is $e = 3$. Then the ciphertext is

$$\mathsf{c} = \mathsf{m}^3 \pmod N.$$

If such a ciphertext is intercepted then the cryptanalyst only needs to find the value for K. In this case we know that K is a solution to the polynomial

$$F(x) = (2^{1024} - 2^{128} + x)^3 - \mathsf{c} \equiv 0 \pmod N.$$

This is a polynomial of degree 3 with a root modulo N of size at most $N^{128/1024} = N^{1/8}$. So Coppersmith's method finds the solution K in polynomial-time.

Example 19.4.1. Let $N = 887355420159847950880463233536 1$ (which is a 103 bit integer) and suppose Bob is sending 10-bit keys K to Alice using the padding scheme $\mathsf{m} = 2^{100} - 2^{10} + K$.

Suppose we have intercepted the ciphertext $\mathsf{c} = 809057455777566200535445549107 6$ and wish to find K. Let $X = 2^{10}$. We write $F(x) = (x + 2^{100} - 2^{10})^3 - \mathsf{c} = x^3 + a_2x^2 + a_1x + a_0$ and define

$$B = \begin{pmatrix} N & 0 & 0 & 0 \\ 0 & NX & 0 & 0 \\ 0 & 0 & NX^2 & 0 \\ a_0 & a_1X & a_2X^2 & X^3 \end{pmatrix}.$$

Performing lattice reduction and taking the first row vector gives the polynomial with factorisation

$$(x - 987)(-920735567540915376297 + 726745175435904508x + 277605904865853x^2).$$

One can verify that the message is $K = 987$.

19.4.2 Factoring $N = pq$ with Partial Knowledge of p

Let $N = pq$ and suppose we are given an approximation $\tilde{p}$ to p such that $p = \tilde{p} + x_0$ where $|x_0| < X$. For example, suppose p is a 2κ-bit prime and $\tilde{p}$ is an integer that has the same κ most significant bits as p (so that $|p - \tilde{p}| < 2^\kappa$). Coppersmith used his ideas to get an algorithm for finding p given N and $\tilde{p}$. Note that Coppersmith originally used a bivariate polynomial method, but we present a simpler version following work of Howgrave-Graham, Boneh, Durfee and others.

The polynomial $F(x) = (x+\tilde{p})$ has a small solution modulo p. The problem is that we don't know p, but we do know a multiple of p (namely, N). The idea is to form a lattice corresponding to polynomials that have a small root modulo p and to apply Coppersmith's method to find this root x_0. Once we have x_0 then we compute p as $\gcd(N, F(x_0))$.

Theorem 19.4.2. *Let $N = pq$ with $p < q < 2p$. Let $0 < \epsilon < 1/4$, and suppose $\tilde{p} \in \mathbb{N}$ is such that $|p-\tilde{p}| \leq \frac{1}{2\sqrt{2}} N^{1/4-\epsilon}$. Then given N and $\tilde{p}$ one can factor N in time polynomial in $\log(N)$ and $1/\epsilon$.*

Proof: Write $F(x) = (x + \tilde{p})$ and note that $\sqrt{N/2} \leq p \leq \sqrt{N}$. Let $X = \lfloor \frac{1}{2\sqrt{2}} N^{1/4-\epsilon} \rfloor$.

We describe the lattice to be used. Let $h \geq 4$ be an integer to be determined later and let $k = 2h$. Consider the $k + 1$ polynomials

$$N^h, N^{h-1}F(x), N^{h-2}F(x)^2, \ldots, NF(x)^{h-1}, F(x)^h, xF(x)^h, \ldots, x^{k-h}F(x)^h.$$

Note that if $p = \tilde{p} + x_0$ and if $G(x)$ is one of these polynomials then $G(x_0) \equiv 0 \pmod{p^h}$.

Consider the lattice corresponding to the above polynomials. More precisely, a basis for the lattice is obtained by taking each polynomial $G(x)$ above and writing the vector of coefficients of the polynomial $G(x)$ as in equation (19.1). The lattice has dimension $k + 1$ and determinant $N^{h(h+1)/2}X^{k(k+1)/2}$.

Applying LLL gives a short vector and, to apply Howgrave-Graham's result, we need $2^{k/4} \det(L)^{1/(k+1)} < p^h/\sqrt{k+1}$. Hence, since $p > (N/2)^{1/2}$, it is sufficient that $\sqrt{k+1}\, 2^{k/4} N^{h(h+1)/(2(k+1))} X^{k/2} < (N/2)^{h/2}$. Re-arranging gives

$$X < N^{h/k - h(h+1)/(k(k+1))} 2^{-h/k} 2^{-1/2} / (k+1)^{1/k}.$$

Since $k \geq 7$ we have $(k+1)^{1/k} = 2^{\log_2(k+1)/k} \leq 2^{1/2}$ and so $1/(k+1)^{1/k} \geq 1/\sqrt{2}$.

Now, since $k = 2h$ we find that the result holds if

$$X < N^{1/2(1-(h+1)/(2h+1))} \tfrac{1}{2\sqrt{2}}.$$

Since $1/2(1-(h+1)/(2h+1)) = 1/4-1/(4(2h+1))$ the result will follow if $1/(4(2h+1)) < \epsilon$. Taking $h \geq \max\{4, 1/(4\epsilon)\}$ is sufficient. □

One can obtain a more general version of Theorem 19.4.2. If $p = N^\alpha$ and $|x| \leq N^\beta$ where $0 < \alpha, \beta < 1$ then, ignoring constants, the required condition in the proof is

$$\frac{h(h+1)}{2} + \frac{\beta k(k+1)}{2} < \alpha h(k+1).$$

Taking $h = \sqrt{\beta}k$ and simplifying gives $\beta < \alpha^2$. The case we have shown is $\alpha = 1/2$ and $\beta < 1/4$. For details see Exercise 19.4.5 or Theorems 6 and 7 of May [410].

Example 19.4.3. Let $N = 16803551$, $\tilde{p} = 2830$ and $X = 10$.

Let $F(x) = (x + \tilde{p})$ and consider the polynomials $N, F(x), xF(x) = (x^2 + \tilde{p}x)$ and $x^2F(x)$, which all have the same small solution x_0 modulo p.

We build the lattice corresponding to these polynomials (with the usual method of converting a polynomial into a row vector). This lattice has basis matrix

$$\begin{pmatrix} N & 0 & 0 & 0 \\ \tilde{p} & X & 0 & 0 \\ 0 & \tilde{p}X & X^2 & 0 \\ 0 & 0 & \tilde{p}X^2 & X^3 \end{pmatrix}.$$

The first row of the output of the LLL algorithm on this matrix is $(105, -1200, 800, 1000)$, which corresponds to the polynomial

$$G(x) = x^3 + 8x^2 - 120x + 105.$$

The polynomial has the root $x = 7$ over $\mathbb{Z}$. We can check that $p = \tilde{p} + 7 = 2837$ is a factor of N.

Exercise 19.4.4. Let $N = 22461580086470571723189523$ and suppose you are given the approximation $\tilde{p} = 2736273600000$ to p, which is correct up to a factor $0 \leq x < X = 50000$. Find the prime factorisation of N using Coppersmith's method.

Exercise 19.4.5. Let $\epsilon > 0$. Let $F(x)$ be a polynomial of degree d such that $F(x_0) \equiv 0 \pmod{M}$ for some $M \mid N$, $M = N^\alpha$ and $|x_0| \leq \frac{1}{2}N^{\alpha^2/d-\epsilon}$. Generalise the proof of Theorem 19.4.2 to show that given $F(x)$ and N one can compute x_0 in time polynomial in $\log(N)$, d and $1/\epsilon$.

Exercise 19.4.6. Coppersmith showed that one can factor N in time polynomial in $\log(N)$ given $\tilde{p}$ such that $|p - \tilde{p}| < N^{1/4}$. Prove this result.

Exercise 19.4.7. Use Coppersmith's method to give an integer factorisation algorithm requiring $\tilde{O}(N^{1/4})$ bit operations. (A factoring algorithm with this complexity was also given in Section 12.5.)

Exercise 19.4.8. Show that the method of this section also works if given $\tilde{p}$ such that $|\tilde{p} - kp| < N^{1/4}$ for some integer k such that $\gcd(k, N) = 1$.

Exercise 19.4.9. Coppersmith also showed that one can factor N in time polynomial in $\log(N)$ given $\tilde{p}$ such that $p \equiv \tilde{p} \pmod{M}$ where $M > N^{1/4}$. Prove this result.

Exercise 19.4.10. Let $N = pq$ with $p \approx q$. Show that if one knows half the high order bits of p then one also knows approximately half the high order bits of q as well.

19.4.3 Factoring $p^r q$

As mentioned in Section 24.1.2, moduli of the form $p^r q$, where p and q are distinct primes and $r \in \mathbb{N}$, can be useful for some applications. When r is large then p is relatively small compared with N and so a natural attack is to try to factor N using the elliptic curve method.

Boneh, Durfee and Howgrave-Graham [79] considered using Coppersmith's method to factor integers of the form $N = p^r q$ when r is large. They observed that if one knows r and an approximation $\tilde{p}$ to p then there is a small root of the polynomial equation

$$F(x) = (\tilde{p} + x)^r \equiv 0 \pmod{p^r}$$

and that p^r is a large factor of N. One can therefore apply the technique of Section 19.4.2

The algorithm is to repeat the above for all $\tilde{p}$ in a suitably chosen set. An analysis of the complexity of the method is given in [79]. It is shown that if $r \geq \log(p)$ then the algorithm runs in polynomial-time and that if $r = \sqrt{\log_2(p)}$ then the algorithm is asymptotically faster than using the elliptic curve method. One specific example mentioned in [79] is that if $p, q \approx 2^{512}$ and $r = 23$ then $N = p^r q$ should be factored more quickly by their method than with the elliptic curve method.

Exercise 19.4.11. Let $N = p^r q$ where $p \approx q$, and so $p \approx N^{1/(r+1)}$. Show that one can factor N in $O(N^{1/(r+1)^2+\epsilon})$ bit operations. In particular, one can factor integers $N = p^2 q$ in roughly $\tilde{O}(N^{1/9})$ bit operations and integers $N = p^3 q$ in roughly $\tilde{O}(N^{1/16})$ bit operations.

When r is small it is believed that moduli of the form $N = p^r q$ are still hard to factor. For 3076 bit moduli, taking $r = 3$ and $p, q \approx 2^{768}$ should be such that the best known attack requires at least 2^{128} bit operations.

Exercise 19.4.12. The integer 876701170324027 is of the form $p^3 q$ where $|p - 5000| < 10$. Use the method of this section to factor N.

19.4.4 Chinese Remaindering with Errors

Boneh [75], building on work of Goldreich, Ron and Sudan [257], used ideas very similar to Coppersmith's method to give an algorithm for the following problem in certain cases.

Definition 19.4.13. Let $X, p_1, \ldots, p_n, r_1, \ldots, r_n \in \mathbb{Z}_{\geq 0}$ be such that $p_1 < p_2 < \cdots < p_n$ and $0 \leq r_i < p_i$ for all $1 \leq i \leq n$. Let $1 \leq e \leq n$ be an integer. The **Chinese remaindering with errors problem** (or **CRT list decoding problem**) is to compute an integer $0 \leq x < X$ (if it exists) such that

$$x \equiv r_i \pmod{p_i}$$

for all but e of the indices $1 \leq i \leq n$.

Note that it is not assumed that the integers p_i are coprime, though in many applications they will be distinct primes or prime powers. Also note that there is not necessarily a solution to the problem (for example, if X and/or e are too small).

Exercise 19.4.14. A naive approach to this problem is to run the Chinese remainder algorithm for all subsets $S \subseteq \{p_1, \ldots, p_n\}$ such that $\#S = (n - e)$. Determine the complexity of this algorithm. What is the input size of a Chinese remainder with errors instance when $0 \leq r_i < p_i$? Show that this algorithm is not polynomial in the input size if $e > \log(n)$.

The basic idea of Boneh's method is to construct a polynomial $F(x) \in \mathbb{Z}[x]$ such that all solutions x to the Chinese remaindering with errors problem instance are roots of $F(x)$ over $\mathbb{Z}$. This is done as follows. Define $P = \prod_{i=1}^{n} p_i$ and let $0 \leq R < P$ be the solution to the Chinese remainder instance (i.e., $R \equiv r_i \pmod{p_i}$ for all $1 \leq i \leq n$). For an integer x define the **amplitude** $\mathrm{amp}(x) = \gcd(P, x - R)$ so that, if the p_i are coprime and S is the set of indices $1 \leq i \leq n$ such that $x \equiv r_i \pmod{p_i}$, then $\mathrm{amp}(x) = \prod_{i \in S} p_i$. Write $F(x) = x - R$. The problem is precisely to find an integer x such that $|x| < X$ and $F(x) \equiv 0 \pmod{M}$ for some large integer $M \mid P$. This is the problem solved by Coppersmith's algorithm in the variant of Exercise 19.4.5. Note that $p_1^n \leq P \leq p_n^n$ and so $n \log(p_1) \leq \log(P) \leq n \log(p_n)$.

Theorem 19.4.15. *Let $X, e, p_1, \ldots, p_n, r_1, \ldots, r_n$ be an instance of the Chinese remainder with errors problem, where $p_1 < p_2 < \cdots < p_n$. Let $P = p_1 \cdots p_n$. There is an algorithm to compute all $x \in \mathbb{Z}$ such that $|x| < X$ and $x \equiv r_i \pmod{p_i}$ for all but e values $1 \leq i \leq n$ as long as*

$$e \leq n - n\frac{\log(p_n)}{\log(p_1)}\sqrt{\log(X)/\log(P)}.$$

The algorithm is polynomial-time in the input size.

Proof: Boneh [75] gives a direct proof, but we follow Section 4.7 of May [411] and derive the result using Exercise 19.4.5.

Let $0 \leq x < X$ be an integer with $M = \mathrm{amp}(x)$ being divisible by at least $n - e$ of the values p_i. We have $n \log(p_1) \leq \log(P) \leq n \log(p_n)$ and $(n-e)\log(p_1) \leq M \leq n \log(p_n)$. Write $M = P^{\beta}$. Then Coppersmith's algorithm finds x if $X < P^{\beta^2}$ in polynomial-time in n and $\log(p_n)$ (note that Exercise 19.4.5 states the result for $X < P^{\beta^2 - \epsilon}$ but we can remove the ϵ using the same ideas as Remark 19.1.13). Hence, it is sufficient to give a bound on e so that $\log(X)/\log(P) < \beta^2$ (i.e., $\beta > \sqrt{\log(X)/\log(P)}$). Now, $\beta = \log(M)/\log(P) \geq (n-e)\log(p_1)/(n\log(p_n))$. Hence, it is sufficient that

$$(n-e)\frac{\log(p_1)}{\log(p_n)} \geq n\sqrt{\log(X)/\log(P)},$$

which is equivalent to the equation in the Theorem. □

For convenience we briefly recall how to perform the computation. One chooses appropriate integers $a, a' \in \mathbb{N}$ and considers the lattice corresponding to the polynomials

$$\begin{aligned} G_i(x) &= P^{a-i}(x-R)^i \quad \text{for } 0 \leq i < a \\ H_i(x) &= (x-R)^a x^i \quad \text{for } 0 \leq i < a' \end{aligned}$$

that, by assumption, have at least one common small root x_0 modulo M^a. Using lattice basis reduction one finds a polynomial $F(x)$ that has small coefficients and that still has the same root x_0 modulo M^a. Applying Theorem 19.1.2 one finds that $F(x_0) = 0$ over $\mathbb{Z}$ if M^a is sufficiently large compared with x_0.

Exercise 19.4.16. Suppose $p_1, \ldots, p_n$ are the first n primes. Show that the above algorithm works when $e \approx n - \sqrt{n \log(X) \log(n)}$. Hence verify that Boneh's algorithm is polynomial-time in situations where the naive algorithm of Exercise 19.4.14 would be superpolynomial-time.

Bleichenbacher and Nguyen [70] discuss a variant of the Chinese remaindering with errors problem (namely, solving $x \equiv r_i \pmod{p_i}$ for small x, where each r_i lies in a set of m possible values) and a related problem in polynomial interpolation. Section 5 of [70] gives some algorithms for this "noisy CRT" problem.

Smooth Integers in Short Intervals

The above methods can be used to find smooth integers in intervals. Let $I = [U, V] = \{x \in \mathbb{Z} : U \leq x \leq V\}$ and suppose we want to find a B-smooth integer $x \in I$ if one exists (i.e., all primes dividing x are at most B). We assume that $V < 2U$.

Exercise 19.4.17. Show that if $V \geq 2U$ then one can compute a power of 2 in $[U, V]$.

A serious problem is that only rather weak results have been proven about smooth integers in short intervals (see Section 4 of Granville [267], Sections 6.2 and 7.2 of Naccache and Shparlinski [450] or Section 15.3). Hence, we cannot expect to be able to prove anything rigorous in this section. On the other hand, it is natural to conjecture that, at least most of the time, the probability that a randomly chosen integer in an short interval $[U, V]$ is B-smooth is roughly equal to the probability that a randomly chosen integer of size V is B-smooth. Multiplying this probability by the length of the interval gives a rough guide to whether it is reasonable to expect a solution (see Remark 15.3.5). Hence, for the remainder of this section, we assume that such an integer x exists. We now sketch how the previous results might be used to find x.

Let $W = (U+V)/2$ and $X = (V-U)/2$ so that $I = [W-X, W+X]$. We seek all $x \in \mathbb{Z}$ such that $|x| \leq X$ and $x \equiv -W \pmod{p_i^{e_i}}$ for certain prime powers where $p_i \leq B$. Then $W + x$ is a potentially smooth integer in the desired interval (we know that $W + x$ has a large smooth factor, but this may not imply that all prime factors of $W + x$ are small if W is very large). One therefore chooses $P = \prod_{i=1}^{l} p_i^{e_i}$ where $p_1, \ldots, p_l$ are the primes up to B and the e_i are suitably chosen exponents (e.g. $e_i = \lceil \log(W)/(\log(B)\log(p_i)) \rceil$). One then applies Boneh's algorithm. The output is an integer with a large common divisor with P (indeed, this is a special case of the approximate GCD problem considered in Section 19.6). Note that this yields rather "dense" numbers, in the sense that they are divisible by most of the first l primes.

Example 19.4.18. Let $P = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 = 232792560$. Let $W = 100000007 = 10^8 + 7$ and $X = 1000000 = 10^6$. We want to find an integer x between $W - X$ and $W + X$ such that x is divisible by most of the prime powers dividing P.

Taking $R = -W$, $a = 4$ and $a' = 3$ in the notation of Theorem 19.4.15 gives the lattice given by the basis matrix

$$\begin{pmatrix} P^4 & 0 & 0 & 0 & 0 & 0 & 0 \\ -RP^3 & P^3X & 0 & 0 & 0 & 0 & 0 \\ R^2P^2 & -2RP^2X & P^2X^2 & 0 & 0 & 0 & 0 \\ -R^3P & 3R^2PX & -3RPX^2 & PX^3 & 0 & 0 & 0 \\ R^4 & -4R^3X & 6R^2X^2 & -4RX^3 & X^4 & 0 & 0 \\ 0 & R^4X & -4R^3X^2 & 6R^2X^3 & -4RX^4 & X^5 & 0 \\ 0 & 0 & R^4X^2 & -4R^3X^3 & 6R^2X^4 & -4RX^5 & X^6 \end{pmatrix}.$$

The polynomial corresponding to the first row of the LLL-reduced basis is

$$F(x) = -7^4(x + 231767)^4$$

giving the solution $x = -231767$. Indeed

$$W - 231767 = 2^4 \cdot 3^3 \cdot 5 \cdot 11 \cdot 13 \cdot 17 \cdot 19.$$

Note that the algorithm does not output $10^8 = 2^8 \cdot 5^8$, since that number does not have a very large gcd with P.

Exercise 19.4.19. Repeat the above example for $W = 150000001 = 1.5 \cdot 10^8 + 1$ and $W = 46558000$.

If this process fails one can make adjustments to the value of P (for example, by changing the exponents e_i). Analysing the probability of success of this approach is an open problem.

19.5 Simultaneous Diophantine Approximation

Let $\alpha \in \mathbb{R}$. It is well-known that the continued fraction algorithm produces a sequence of rational numbers p/q such that $|\alpha - p/q| < 1/q^2$. This is the subject of **Diophantine approximation**; see Section 1.1 of Lovász [395] for background and discussion. We now define a natural and important generalisation of this problem.

Definition 19.5.1. Let $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$ and let $\epsilon > 0$. Let $Q \in \mathbb{N}$ be such that $Q \geq \epsilon^{-n}$. The **simultaneous Diophantine approximation problem** is to find $q, p_1, \ldots, p_n \in \mathbb{Z}$ such that $0 < q \leq Q$ and

$$|\alpha_i - p_i/q| \leq \epsilon/q \tag{19.4}$$

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

A theorem of Dirichlet mentioned in Section 1.1 of [395] and Section 17.3 of [238] shows that there is a solution satisfying the constraints in Definition 19.5.1.

Exercise 19.5.2. Let $\epsilon \geq 1/2$. Prove that integers $p_1, \ldots, p_n$ satisfying equation (19.4) exist for any n and q.

A major application of lattice reduction is to give an algorithm to compute the integers $(q, p_1, \ldots, p_n)$ in Definition 19.5.1. In practice the real numbers $\alpha_1, \ldots, \alpha_n$ are given to some decimal precision (and so are rational numbers with coefficients of some size). The size of an instance of the simultaneous Diophantine approximation is the sum of the bit lengths of the numerator and denominator of the given approximations to the α_i, together with the bit length of the representation of ϵ and Q. Let X be a bound on the absolute value of all numerators and denominators of the α_i. The computational task is to find a solution $(q, p_1, \ldots, p_n)$ in time that is polynomial in n, $\log(X)$, $\log(1/\epsilon)$ and $\log(Q)$.

Theorem 19.5.3. *Let $\alpha_1, \ldots, \alpha_n \in \mathbb{Q}$ be given as rational numbers with numerator and denominator bounded in absolute value by X. Let $0 < \epsilon < 1$. One can compute in polynomial-time integers $(q, p_1, \ldots, p_n)$ such that $0 < q < 2^{n(n+1)/4}\epsilon^{-(n+1)}$ and $|\alpha_i - p_i/q| \leq \epsilon/q$ for all $1 \leq i \leq n$.*

Proof: Let $Q = 2^{n(n+1)/4}\epsilon^{-n}$ and consider the lattice $L \subseteq \mathbb{Q}^{n+1}$ with basis matrix

$$\begin{pmatrix} \epsilon/Q & \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ 0 & -1 & 0 & \cdots & 0 \\ 0 & 0 & -1 & & \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & & \cdots & -1 \end{pmatrix}. \tag{19.5}$$

The dimension is $n+1$ and the determinant is $\epsilon/Q = 2^{-n(n+1)/4}\epsilon^{n+1}$. Every vector in the lattice is of the form $(q\epsilon/Q, q\alpha_1 - p_1, q\alpha_2 - p_2, \ldots, q\alpha_n - p_n)$. The entries of the lattice are ratios of integers with absolute value bounded by $\max\{X, 2^{n(n+1)/4}/\epsilon^{n+1}\}$.

Note that the lattice L does not have a basis with entries in $\mathbb{Z}$, but rather in $\mathbb{Q}$. By Remark 17.5.5 the LLL algorithm applied to L runs in $O(n^6 \max\{n\log(X), n^2 + n\log(1/\epsilon)\}^3)$ bit operations (which is polynomial in the input size) and outputs a non-zero vector $\underline{v} = (q\epsilon/Q, q\alpha_1 - p_1, \ldots, q\alpha_n - p_n)$ such that

$$\|\underline{v}\| \leq 2^{n/4}\det(L)^{1/(n+1)} = 2^{n/4}2^{-n/4}\epsilon = \epsilon < 1.$$

If $q = 0$ then $\underline{v} = (0, -p_1, \ldots, -p_n)$ with some $p_i \neq 0$ and so $\|\underline{v}\| \geq 1$, and so $q \neq 0$. Without loss of generality, $q > 0$. Since $\|\underline{v}\|_\infty \leq \|\underline{v}\|$ it follows that $q\epsilon/Q \leq \epsilon < 1$ and so $0 < q < Q/\epsilon = 2^{n(n+1)/4}\epsilon^{-(n+1)}$. Similarly, $|q\alpha_i - p_i| < \epsilon$ for all $1 \leq i \leq n$. □

Exercise 19.5.4. Let $\alpha_1 = 1.555111$, $\alpha_2 = 0.771111$ and $\alpha_3 = 0.333333$. Let $\epsilon = 0.01$ and $Q = 10^6$. Use the method of this section to find a good simultaneous rational approximation to these numbers.

See Section 17.3 of [238] for more details and references.

19.6 Approximate Integer Greatest Common Divisors

The basic problem is the following. Suppose positive integers a and b exist such that $d = \gcd(a, b)$ is "large". Suppose that one is not given a and b, but only approximations $\tilde{a}, \tilde{b}$ to them. The problem is to find d, a and b. One issue is that there can be surprisingly many solutions to the problem (see Example 19.6.4), so it may not be feasible to compute all solutions for certain parameters. On the other hand, in the case $\tilde{b} = b$ (i.e., one of the values is known exactly, which often happens in practice) then there are relatively few solutions.

Howgrave-Graham [297] has considered these problems and has given algorithms that apply in various situations. We present one of the basic ideas. Let $a = \tilde{a} + x$ and $b = \tilde{b} + y$. Suppose $\tilde{a} < \tilde{b}$ and define $q_a = a/d$ and $q_b = b/d$. Then, since $q_a/q_b = a/b$, we have

$$\frac{\tilde{a}}{\tilde{b}} - \frac{q_a}{q_b} = \frac{q_a y - q_b x}{\tilde{b} q_b}. \tag{19.6}$$

If the right hand side of equation (19.6) is small then performing Euclid's algorithm on $\tilde{a}/\tilde{b}$ gives a sequence of possible values for q_a/q_b. For each such value one can compute

$$\lfloor \tilde{b}/q_b \rceil = \lfloor (dq_b - y)/q_b \rceil = d + \lfloor -y/q_b \rceil.$$

If $|y| < \frac{1}{2}q_b$ then one has computed d exactly and can solve $\tilde{a} + x \equiv \tilde{b} + y \equiv 0 \pmod{d}$. Note that one must use the basic extended Euclidean algorithm, rather than the improved method using negative remainders as in Algorithm 1.

Exercise 19.6.1. Show that if $a < b < \tilde{b}$, $b^{2/3} < d < 2b^{2/3}$ and $|x|, |y| < \frac{1}{4}b^{1/3}$ then the above method finds d, a and b.

Exercise 19.6.2. Let the notation be as above. Suppose $|x|, |y| < \tilde{b}^\beta$ and $d = \tilde{b}^\alpha$. Explain why it is natural to assume $\alpha > \beta$. Show that the above method succeeds if (ignoring constant factors) $\beta < -1 + 2\alpha$ and $\beta < 1 - \alpha$

Exercise 19.6.3. Re-formulate this method in terms of finding a short vector in a 2×2 matrix. Derive the same conditions on α and β as in Exercise 19.6.2.

Example 19.6.4. Let $\tilde{a} = 617283157$ and $\tilde{b} = 630864082$. The first few convergents q_a/q_b to $\tilde{a}/\tilde{b}$ are $1, 45/46, 91/93, 409/418, 500/511, 1409/1440$ and $1909/1951$. Computing approximations to $\tilde{a}/q_a$ and $\tilde{b}/q_b$ for these values (except the first) gives the following table.

$\tilde{a}/q_a$	13717403.5	6783331.4	1509249.8	1234566.3	438100.2	323354.2
$\tilde{b}/q_b$	13714436.6	6783484.8	1509244.2	1234567.7	438100.1	323354.2

Any values around these numbers can be used as a guess for d. For example, taking $d = 13717403$ one finds $\tilde{a} - 22 \equiv \tilde{b} + 136456 \equiv 0 \pmod{d}$, which is a not particularly good solution.

The four values $d_1 = 1234566$, $d_2 = 1234567$, $d_3 = 438100$ and $d_4 = 323354$ lead to the solutions $\tilde{a} - 157 \equiv \tilde{b} - 856 \equiv 0 \pmod{d_1}$, $\tilde{a} + 343 \equiv \tilde{b} - 345 \equiv 0 \pmod{d_2}$, $\tilde{a} - 257 \equiv \tilde{b} - 82 \equiv 0 \pmod{d_3}$ and $\tilde{a} - 371 \equiv \tilde{b} - 428 \equiv 0 \pmod{d_4}$.

Howgrave-Graham gives a more general method for solving the problem that does not require such a strict condition on the size of y. The result relies on heuristic assumptions about Coppersmith's method for bivariate integer polynomials. We state this result as Conjecture 19.6.5.

Conjecture 19.6.5. *(Algorithm 14 and Section 4 of [297]) Let $0 < \alpha < 2/3$ and $\beta \leq 1 - \alpha/2 - \sqrt{1 - \alpha - \alpha^2/2}$. There is a polynomial-time algorithm that takes as input $\tilde{a} < \tilde{b}$ and outputs all integers $d > \tilde{b}^\alpha$ such that there exist integers x, y with $|x|, |y| < \tilde{b}^\beta$ and $d \mid (\tilde{a} + x)$ and $d \mid (\tilde{b} + y)$.*

Exercise 19.6.6. Let $\tilde{a}, \tilde{b}, X, Y \in \mathbb{N}$ be given with $X < \tilde{a} < \tilde{b}$. Give a brute force algorithm to output all $d > Y$ such that there exist $x, y \in \mathbb{Z}$ with $|x|, |y| \leq X$ and $d = \gcd(\tilde{a} + x, \tilde{b} + y)$. Show that the complexity of this algorithm is $O(X^2 \log(\tilde{b})^2)$ bit operations.

We now mention the case when $\tilde{b} = b$ (in other words, b is known exactly). The natural approach is to consider the polynomial $F(x) = \tilde{a} + x$, which has a small solution to the equation $F(x) \equiv 0 \pmod{d}$ for some $d \mid b$. Howgrave-Graham applies the method used in Section 19.4.2 to solve this problem.

Theorem 19.6.7. *(Algorithm 12 and Section 3 of [297]) Let $0 < \alpha < 1$ and $\beta < \alpha^2$. There is a polynomial-time algorithm that takes as input $\tilde{a}, b$ and outputs all integers $d > b^\alpha$ such that there exists an integer x with $|x| < b^\beta$ and $d \mid (\tilde{a} + x)$ and $d \mid b$.*

19.7 Learning with Errors

The learning with errors problem was proposed by Regev. There is a large literature on this problem; we refer to Micciancio and Regev [423] and Regev [496] for background and references.

Definition 19.7.1. Let $q \in \mathbb{N}$ (typically prime), $\sigma \in \mathbb{R}_{>0}$, and $n, m \in \mathbb{N}$ with $m > n$.[1] Let $\underline{s} \in (\mathbb{Z}/q\mathbb{Z})^n$. The **LWE distribution** is the distribution on $(\mathbb{Z}/q\mathbb{Z})^{m \times n} \times (\mathbb{Z}/q\mathbb{Z})^m$ corresponding to choosing uniformly at random an $m \times n$ matrix A with entries in $\mathbb{Z}/q\mathbb{Z}$ and a length m vector

$$\underline{c} \equiv A\underline{s} + \underline{e} \pmod{q}$$

[1] For theoretical applications one should not assume a fixed number m of rows for A. Instead, the attacker is given an oracle that outputs pairs $(\underline{a}, c)$ where $\underline{a}$ is a row of A and $c = \underline{a}\,\underline{s} + e \pmod{q}$.

where the vector $\underline{e}$ has entries chosen independently from a discretised normal distribution[2] on $\mathbb{Z}$ with mean 0 and standard deviation σ. The **learning with errors** problem (**LWE**) is: Given $(A, \underline{c})$ drawn from the LWE distribution, to compute the vector $\underline{s}$. The **decision learning with errors** problem (**DLWE**) is: Given A as above and a vector $\underline{c} \in (\mathbb{Z}/q\mathbb{Z})^m$, to determine whether $(A, \underline{c})$ is drawn from the uniform distribution, or the LWE distribution.

It is necessary to argue that LWE is well-defined since, for any choice $\underline{s}'$, the value $\underline{c} - A\underline{s}' \pmod q$ is a possible choice for $\underline{e}$. But, when m is sufficiently large, one value for $\underline{s}$ is much more likely to have been used than any of the others. Hence, LWE is a maximum likelihood problem. Similarly, DLWE is well-defined when m is sufficiently large: if $\underline{c}$ is chosen uniformly at random and independent of A then there is not likely to be a choice for $\underline{s}$ such that $\underline{c} - A\underline{s} \pmod q$ is significantly smaller than the other values $\underline{c} - A\underline{s}' \pmod q$. We do not make these arguments precise. It follows that m must be significantly larger than n for these problems to be meaningful. It is also clear that increasing m (but keeping n fixed) does not make the LWE problem harder.

We refer to [423] and [496] for surveys of cryptographic applications of LWE and reductions, from computational problems in lattices that are believed to be hard, to LWE. Note that the values m, q and σ in an LWE instance are usually determined by constraints coming from the cryptographic application, while n is the main security parameter.

Example 19.7.2. Table 3 of Micciancio and Regev [423] suggests the parameters

$$(n, m, q, \sigma) = (233, 4536, 32749, 2.8).$$

Lindner and Peikert [390] suggest (using Figure 4 and the condition $m \geq 2n + \ell$ with $\ell = 128$)

$$(n, m, q, \sigma) = (256, 640, 4093, 3.3).$$

Exercise 19.7.3. Show that if one can determine $\underline{e}$ then one can solve LWE efficiently.

Exercise 19.7.4.★ Show that, when q is prime, LWE $\leq_R$ DLWE. Show that DLWE $\leq_R$ LWE.

We now briefly sketch two lattice attacks on LWE. These attacks can be avoided by taking appropriate parameters. For other attacks on LWE see [496].

Example 19.7.5. (Lattice attack on DLWE using short vectors in kernel lattice modulo q.) Suppose one can find a short vector $\underline{w}$ in the lattice

$$\{\underline{w} \in \mathbb{Z}^m : \underline{w}A \equiv \underline{0} \pmod q\}.$$

Then $\underline{w}\,\underline{c} = \underline{w}A\underline{s} + \underline{w}\,\underline{e} \equiv \underline{w}\,\underline{e} \pmod q$. If $\underline{w}$ is short enough then one might expect that $\underline{w}\,\underline{e}$ is a small integer. On the other hand, if $\underline{c}$ is independent of A then $\underline{w}\,\underline{c} \pmod q$ is a random integer modulo q. Hence, one might be able to distinguish the LWE distribution from the uniform distribution using short enough vectors $\underline{w}$.

Note that one is not obliged to use all the rows of A in this attack, and so one can replace m by a much smaller value m'. For analysis of the best value for m', and for parameters that resist this attack, see Section 5.4.1 (especially equation (10)) of [423].

Example 19.7.6. (Reducing LWE to bounded distance decoding (BDD) or CVP.) We now consider a natural approach to solving LWE using lattices. Since we always use row

[2] In other words, the probability that e_i is equal to $x \in \mathbb{Z}$ is proportional to $e^{-x^2/(2\sigma^2)}$.

lattices, it is appropriate to take the transpose of LWE. Hence, suppose $\underline{c}, \underline{s}$ and $\underline{e}$ are row vectors (of lengths m, n and m respectively) such that $\underline{c} = \underline{s}A^T + \underline{e} \pmod{q}$.

Consider the lattice

$$L = \left\{ \underline{v} \in \mathbb{Z}^m : \underline{v} \equiv \underline{u}A^T \pmod{q} \text{ for some } \underline{u} \in \mathbb{Z}^n \right\}.$$

Then L has rank m and a basis matrix for it is computed by taking the (row) Hermite normal form of the $(n+m) \times m$ matrix

$$\begin{pmatrix} A^T \\ qI_m \end{pmatrix}$$

where I_m is an $m \times m$ identity matrix. One then tries to find an element $\underline{v}$ of L that is close to $\underline{c}$. Hopefully, $\underline{v} = \underline{c} - \underline{e} \equiv \underline{s}A^T \pmod{q}$. For usual LWE parameters we have that there is a unique $\underline{v} \in L$ that is very close to $\underline{c}$, and so the problem matches the bounded distance decoding problem.

One can perform lattice basis reduction and apply the nearest plane algorithm. For improved methods and experimental results see Lindner and Peikert [390]. As in Example 19.7.5 one can work with a subset of m' rows of A; see Section 5.1 of [390] for details.

19.8 Further Applications of Lattice Reduction

There are a number of other applications of lattices in cryptography. We briefly list some of them.

- The improvement by Boneh and Durfee of Wiener's attack on small private exponent RSA. This is briefly mentioned in Section 24.5.1.
- Solving the hidden number problem in finite fields and its applications to bit security of Diffie-Hellman key exchange. See Section 21.7.1.
- The attack by Howgrave-Graham and Smart on digital signature schemes in finite fields when there is partial information available about the random nonces. See Section 22.3.
- The deterministic reduction by Coron and May from knowing $\varphi(N)$ to factoring N. This is briefly mentioned in Section 24.1.3.