Chapter 23 Lattice Methods

Lattices have become an important tool for the cryptanalyst. In this chapter, we give a sampling of some of the techniques. In particular, we use lattice reduction techniques to attack RSA in certain cases. Also, we describe the NTRU public key system and show how it relates to lattices. For a more detailed survey of cryptographic applications of lattices, see [Nguyen-Stern].

23.1 Lattices

Let $v_1, \ldots, v_n$ be linearly independent vectors in n-dimensional real space $\mathbf{R}^n$. This means that every n-dimensional real vector v can be written in the form

$$v = a_1 v_1 + \cdots + a_n v_n$$

with real numbers $a_1, \ldots, a_n$ that are uniquely determined by v. The **lattice** generated by $v_1, \ldots, v_n$ is the set of vectors of the form

$$m_1 v_1 + \cdots + m_n v_n$$

where $m_1, \ldots, m_n$ are integers. The set $\{v_1, \ldots, v_n\}$ is called a **basis** of the lattice. A lattice has infinitely many possible bases. For example, suppose $\{v_1, v_2\}$ is a basis of a lattice. Let k be an integer and let $w_1 = v_1 + kv_2$ and $w_2 = v_2$. Then $\{w_1, w_2\}$ is also a basis of the lattice: Any vector of the form $m_1 v_1 + m_2 v_2$ can be written as $m_1' w_1 + m_2' w_2$ with $m_1{}' = m_1$ and $m_2' = m_2 - km_1$, and similarly any integer linear combination of w_1 and w_2 can be written as an integer linear combination of v_1 and v_2.

Example

Let $v_1 = (1, 0)$ and $v_2 = (0, 1)$. The lattice generated by v_1 and v_2 is the set of all pairs (x, y) with x, y integers. Another basis for this lattice is $\{(1, 5),\ (0, 1)\}$. A third basis is $\{(5, 16),\ (6, 19)\}$. More generally, if $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is a matrix with determinant ± 1, then $\{(a, b),\ (c, d)\}$ is a basis of this lattice (Exercise 4).

The length of a vector $v = (x_1, \ldots, x_n)$ is

$$\| v \| = \left(x_1^2 + \cdots + x_n^2\right)^{1/2}.$$

Many problems can be related to finding a shortest nonzero vector in a lattice. In general, the **shortest vector problem** is hard to solve, especially when the dimension of the lattice is large. In the following section, we give some methods that work well in small dimensions.

Example

A shortest vector in the lattice generated by

$$(31, 59) \text{ and } (37, 70)$$

is $(3, \ -1)$ (another shortest vector is $(-3, 1)$). How do we find this vector? This is the subject of the next section. For the moment, we verify that $(3, \ -1)$ is in the lattice by writing

$$(3, \ -1) = -19\,(31, 59) + 16\,(37, 70).$$

In fact, $\{(3, \ -1), \ (1, 4)\}$ is a basis of the lattice. For most purposes, this latter basis is much easier to work with than the original basis since the two vectors $(3, \ -1)$ and $(1, 4)$ are almost orthogonal (their dot product is $(3, \ -1) \cdot (1, 4) = -1$, which is small). In contrast, the two vectors of the original basis are nearly parallel and have a very large dot product. The methods of the next section show how to replace a basis of a lattice with a new basis whose vectors are almost orthogonal.

23.2 Lattice Reduction

23.2.1 Two-Dimensional Lattices

Let v_1, v_2 form the basis of a two-dimensional lattice. Our first goal is to replace this basis with what will be called a reduced basis.

If $\| v_1 \| > \| v_2 \|$, then swap v_1 and v_2, so we may assume that $\| v_1 \| \leq \| v_2 \|$. Ideally, we would like to replace v_2 with a vector v_2^* perpendicular to v_1. As in the Gram-Schmidt process from linear algebra, the vector

$$v_2^* = v_2 - \frac{v_1 \cdot v_2}{v_1 \cdot v_1} v_1$$

(23.1)

is perpendicular to v_1. But this vector might not lie in the lattice. Instead, let t be the closest integer to $(v_1 \cdot v_2)/(v_1 \cdot v_1)$ (for definiteness, take 0 to be the closest integer to $\pm\frac{1}{2}$, and ± 1 to be closest to $\pm\frac{3}{2}$, etc.). Then we replace the basis $\{v_1, \ v_2\}$ with the basis

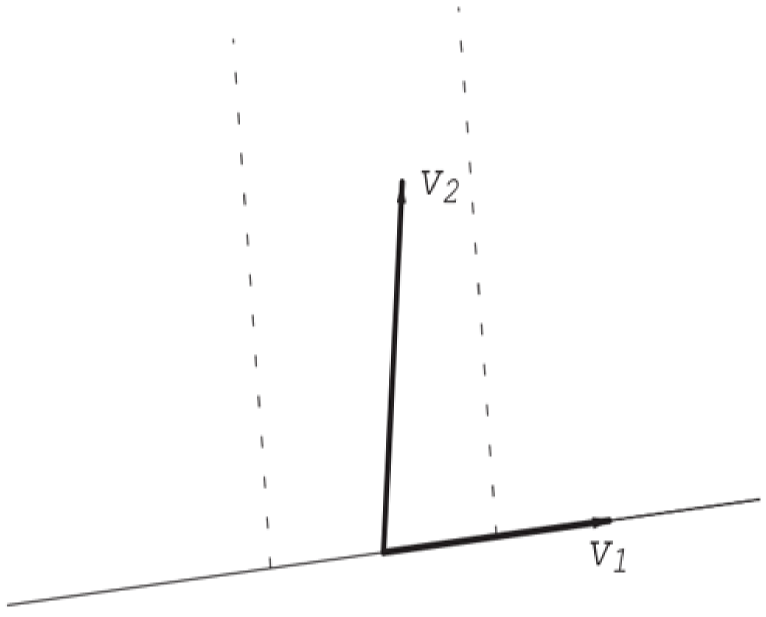
$$\{v_1, \ v_2 - tv_1\}.$$

We then repeat the process with this new basis.

We say that the basis $\{v_1, \ v_2\}$ is **reduced** if

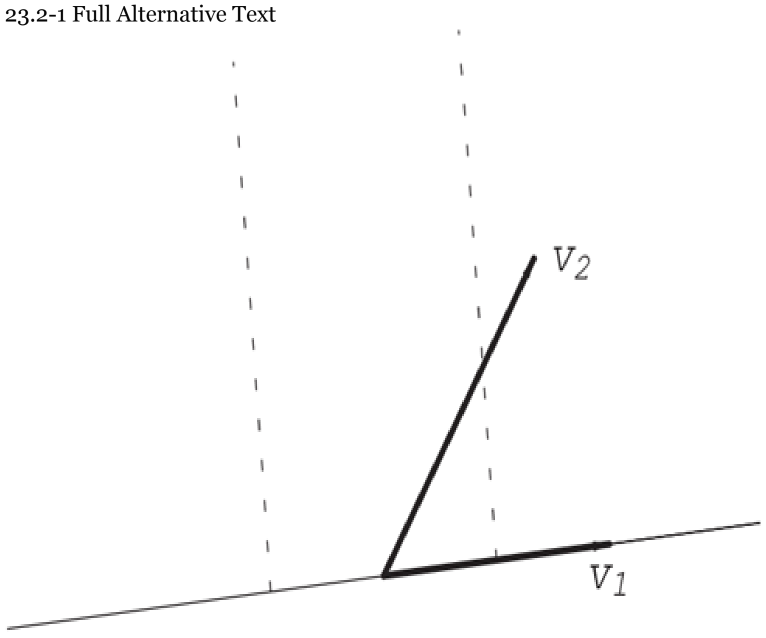
$$\| v_1 \| \leq \| v_2 \| \text{ and } -\frac{1}{2} \leq \frac{v_1 \cdot v_2}{v_1 \cdot v_1} \leq \frac{1}{2}.$$

The above reduction process stops exactly when we obtain a reduced basis, since this means that $t = 0$.



A reduced basis

23.2-1 Full Alternative Text



A nonreduced basis

23.2-2 Full Alternative Text

In the figures, the first basis is reduced because v_2 is longer than v_1 and the projection of v_2 onto v_1 is less than half the length of v_1. The second basis is nonreduced because the projection of v_2 onto v_1 is too long. It is easy to see that a basis $\{v_1,\, v_2\}$ is reduced when v_2 is at least as long as v_1 and v_2 lies within the dotted lines of the figures.

Example

Let's start with $v_1 = (31, 59)$ and $v_2 = (37, 70)$. We have $\| v_1 \| < \| v_2 \|$, so we do not swap the two vectors. Since

$$\frac{v_1 \cdot v_2}{v_1 \cdot v_1} = \frac{5277}{4442},$$

we take $t = 1$. The new basis is

$$v_1^{'} = v_1 = (31, 59) \quad \text{and} \quad v_2^{'} = v_2 - v_1 = (6, 11).$$

Swap $v_1^{'}$ and $v_2^{'}$ and rename the vectors to obtain a basis

$$v_1^{''} = (6, 11) \quad \text{and} \quad v_2^{''} = (31, 59).$$

We have

$$\frac{v_1^{''} \cdot v_2^{''}}{v_1^{''} \cdot v_1^{''}} = \frac{835}{157},$$

so we take $t = 5$. This yields vectors

$$(6, 11) \quad \text{and} \quad (1, 4) = (31, 59) - 5 \cdot (6, 11).$$

Swap these and name them $v_1^{(3)} = (1, 4)$ and $v_2^{(3)} = (6, 11)$. We have

$$\frac{v_1^{(3)} \cdot v_2^{(3)}}{v_1^{(3)} \cdot v_1^{(3)}} = \frac{50}{17},$$

so $t = 3$. This yields, after a swap,

$$v_1^r = (3, -1) \quad \text{and} \quad v_2^r = (1, 4).$$

Since $\| v_1^r \| \leq \| v_2^r \|$ and

$$\frac{v_1^r \cdot v_2^r}{v_1^r \cdot v_1^r} = -\frac{1}{10},$$

the basis $\{v_1^r,\ v_2^r\}$ is reduced.

A natural question is whether this process always produces a reduced basis. The answer is yes, as we prove in the following theorem. Moreover, the first vector in the reduced basis is a shortest vector for the lattice.

We summarize the discussion in the following.

Theorem

Let $\{v_1,\ v_2\}$ be a basis for a two-dimensional lattice in $\mathbf{R}^2$. Perform the following algorithm:

1. If $\| v_1 \| > \| v_2 \|$, swap v_1 and v_2 so that $\| v_1 \| \le \| v_2 \|$.
2. Let t be the closest integer to $(v_1 \cdot v_2)/(v_1 \cdot v_1)$.
3. If $t = 0$, stop. If $t \neq 0$, replace v_2 with $v_2 - tv_1$ and return to step 1.

The algorithm stops in finite time and yields a reduced basis $\{v_1^r,\ v_2^r\}$ of the lattice. The vector v_1^r is a shortest nonzero vector for the lattice.

Proof

First we prove that the algorithm eventually stops. As in Equation 23.1, let $\mu = (v_1 \cdot v_2)/(v_1 \cdot v_1)$ and let $v_2^* = v_2 - \mu v_1$. Then

$$v_2 - tv_1 = v_2^* + (\mu - t)v_1.$$

Since v_1 and v_2^* are orthogonal, the Pythagorean theorem yields

$$\| v_2 - tv_1 \|^2 = \| v_2^* \|^2 + \| (\mu - t)v_1 \|^2 = \| v_2^* \|^2 + (\mu - t)^2 \| v_1 \|^2 .$$

Also, since $v_2 = v_2^* + \mu v_1$, and again since v_1 and v_2^* are orthogonal,

$$\| v_2 \|^2 = \| v_2^* \|^2 + \mu^2 \| v_1 \|^2 .$$

Note that if $-1/2 \leq \mu \leq 1/2$ then $t = 0$ and $\mu - t = \mu$. Otherwise, $|\mu - t| \leq 1/2t|\mu|$. Therefore, if $t \neq 0$, we have $|\mu - t| < |\mu|$, which implies that

$$\| v_2 - tv_1 \|^2 = \| v_2^* \|^2 + (\mu - t)^2 \| v_1 \|^2 < \| v_2^* \|^2 + \mu^2 \| v_1 \|^2 = \| v_2 \|^2 .$$

Therefore, if the process continues forever without yielding a reduced basis, then the lengths of the vectors decrease indefinitely. However, there are only finitely many vectors in the lattice that are shorter than the original basis vector v_2. Therefore, the lengths cannot decrease forever, and a reduced basis must be found eventually.

To prove that the vector v_1 in a reduced basis is a shortest nonzero vector for the lattice, let $av_1 + bv_2$ be any nonzero vector in the lattice, where a and b are integers. Then

$$\| av_1 + bv_2 \|^2 = (av_1 + bv_2) \cdot (av_1 + bv_2) = a^2 \| v_1 \|^2 + b^2 \| v_2 \|^2 + 2ab\, v_1 \cdot v_2.$$

Because $\{v_1, v_2\}$ is reduced,

$$-\frac{1}{2} v_1 \cdot v_1 \leq v_1 \cdot v_2 \leq \frac{1}{2} v_1 \cdot v_1,$$

which implies that $2ab\, v_1 \cdot v_2 \geq -|ab| \; \| v_1 \|^2$. Therefore,

$$\begin{aligned} \|av_1 + bv_2\|^2 &= (av_1 + bv_2) \cdot (av_1 + bv_2) \\ &= a^2 \|v_1\|^2 + 2abv_1 \cdot v_2 + b^2 \|v_2\|^2 \\ &\geq a^2 \|v_1\|^2 - |ab| \|v_1\|^2 + b^2 \|v_2\|^2 \\ &\geq a^2 \|v_1\|^2 - |ab| \|v_1\|^2 + b^2 \|v_1\|^2, \end{aligned}$$

since $\| v_2 \|^2 \geq \| v_1 \|^2$ by assumption. Therefore,

$$\| av_1 + bv_2 \|^2 \geq (a^2 - |ab| + b^2) \| v_1 \|^2 .$$

But $a^2 - |ab| + b^2$ is an integer. Writing it as $(|a| - \frac{1}{2}|b|)^2 + \frac{1}{4}|b|^2$, we see that it is nonnegative, and it equals 0 if and only if $a = b = 0$. Since $av_1 + bv_2 \neq 0$, we must have $a^2 - |ab| + b^2 \geq 1$. Therefore,

$$\| av_1 + bv_2 \|^2 \geq \| v_1 \|^2 ,$$

so v_1 is a shortest nonzero vector.

23.2.2 The LLL algorithm

Lattice reduction in dimensions higher than two is much more difficult. One of the most successful algorithms was invented by A. Lenstra, H. Lenstra, and L. Lovász and is called the *LLL* algorithm. In many problems, a short vector is needed, and it is not necessary that the vector be the shortest. The *LLL* algorithm takes this approach and looks for short vectors that are almost as short as possible. This modified approach makes the algorithm run very quickly (in what is known as polynomial time). The algorithm performs calculations similar to those in the two-dimensional case, but the steps are more technical, so we omit details, which can be found in [Cohen], for example. The result is the following.

Theorem

Let L be the n-dimensional lattice generated by $v_1, \dots, v_n$ in $\mathbf{R}^n$. Define the determinant of the lattice to be

$$D = |\det(v_1, ..., v_n)|.$$

(This can be shown to be independent of the choice of basis. It is the volume of the parallelepiped spanned by $v_1, \dots, v_n$.) Let λ be the length of a shortest nonzero vector in L. The *LLL* algorithm produces a basis $\{b_1, \dots, b_n\}$ of L satisfying

1. $\| b_1 \| \leq 2^{(n-1)/4} D^{1/n}$
2. $\| b_1 \| \leq 2^{(n-1)/2} \lambda$
3. $\| b_1 \| \| b_2 \| \cdots \| b_n \| \leq 2^{n(n-1)/4} D.$

Statement (2) says that b_1 is close to being a shortest vector, at least when the dimension n is small. Statement (3) says that the new basis vectors are in some sense close to being orthogonal. More precisely, if the vectors $b_1, \ldots, b_n$ are orthogonal, then the volume D equals the product $\| b_1 \| \| b_2 \| \cdots \| b_n \|$. The fact that this product is no more than $2^{n(n-1)/4}$ times D says that the vectors are mostly close to orthogonal.

The running time of the *LLL* algorithm is less than a constant times $n^6 \log^3 B$, where n is the dimension and B is a bound on the lengths of the original basis vectors. In practice it is much faster than this bound. This estimate shows that the running time is quite good with respect to the size of the vectors, but potentially not efficient when the dimension gets large.

Example

Let's consider the lattice generated by (31, 59) and (37, 70), which we considered earlier when looking at the two-dimensional algorithm. The *LLL* algorithm yields the same result, namely $b_1 = (3, -1)$ and $b_2 = (1, 4)$. We have $D = 13$ and $\lambda = \sqrt{10}$ (given by $\| (3, -1) \|$, for example). The statements of the theorem are

1. $\| b_1 \| = \sqrt{10} \le 2^{1/4}\sqrt{13}$
2. $\| b_1 \| = \sqrt{10} \le 2^{1/2}\sqrt{10}$
3. $\| b_1 \| \| b_2 \| = \sqrt{10}\sqrt{17} \le 2^{1/2} 13.$

23.3 An Attack on RSA

Alice wants to send Bob a message of the form

$$The\, answer\, is\ **$$

or

$$The\, password\, for\, your\, new\, account\, is\ ********.$$

In these cases, the message is of the form

$$m = B + x,\ \text{ where } B \text{ is fixed and } |x| \leq Y$$

for some integer Y. We'll present an attack that works when the encryption exponent is small.

Suppose Bob has public RSA key $(n,\, e) = (n,\, 3)$. Then the ciphertext is

$$c \equiv (B + x)^3 \pmod n.$$

We assume that Eve knows B, Y, and n, so she only needs to find x. She forms the polynomial

$$\begin{aligned} f(T) &= (B+T)^3 - c = T^3 + 3BT^2 + 3B^2T + B^3 - c \\ &\equiv T^3 + a_2T^2 + a_1T + a_0 \pmod n. \end{aligned}$$

Eve is looking for $|x| \leq Y$ such that $f(x) \equiv 0 \pmod n$. In other words, she is looking for a small solution to a polynomial congruence $f(T) \equiv 0 \pmod n$.

Eve applies the *LLL* algorithm to the lattice generated by the vectors

$$\begin{gathered} v_1 = (n,\, 0,\, 0,\, 0), \quad v_2 = (0,\, Yn,\, 0,\, 0), \quad v_3 = (0,\, 0,\, Y^2n,\, 0), \\ v_4 = (a_0,\, a_1Y,\, a_2Y^2,\, Y^3). \end{gathered}$$

This yields a new basis $b_1,\ \ldots,\ b_4$, but we need only b_1. The theorem in Subsection 23.2.2 tells us that

$$\|\, b_1 \,\| \quad \leq \quad 2^{3/4} \det\left(v_1,\ \ldots,\ v_4\right)^{1/4}$$

(23.2)

$$= \quad 2^{3/4}(n^3Y^6)^{1/4} = 2^{3/4}n^{3/4}Y^{3/2}.$$

(23.3)

We can write

$$b_1 = c_1v_1 + \cdots + c_4v_4 = (e_0, Ye_1, Y^2e_2, Y^3e_3)$$

with integers c_i and with

$$\begin{aligned} e_0 &= c_1n + c_4a_0 \\ e_1 &= c_2n + c_4a_1 \\ e_2 &= c_3n + c_4a_2 \\ e_3 &= c_4. \end{aligned}$$

It is easy to see that

$$e_i \equiv c_4a_i \pmod n,\ 0 \le i \le 2.$$

Form the polynomial

$$g(T) = e_3T^3 + e_2T^2 + e_1T + e_0.$$

Then, since the integer x satisfies $f(x) \equiv 0 \pmod n$ and since the coefficients of $c_4f(T)$ and $g(T)$ are congruent mod n,

$$0 \equiv c_4f(x) \equiv g(x) \pmod n.$$

Assume now that

$$Y < 2^{-7/6}n^{1/6}.$$

(23.4)

Then

$$\begin{aligned} |g(x)| &\le |e_0| + |e_1x| + |e_2x^2| + |e_3x^3| \\ &\le |e_0| + |e_1|Y + |e_2|Y^2 + |e_3|Y^3 \\ &= (1,\ 1,\ 1,\ 1) \cdot \left(|e_0|,\ |e_1Y|,\ |e_2Y^2|,\ |e_3Y^3|\right) \\ &\le \|\ (1,\ 1,\ 1,\ 1)\ \|\ \left(\left(|e_0|,\ \ldots,\ |e_3Y^3|\right)\right) \\ &= 2\ \|\ b_1\ \|, \end{aligned}$$

where the last inequality used the Cauchy-Schwarz inequality for dot products (that is, $v\ \cdot\ w \le \|\ v\ \|\ \|\ w\ \|$). Since, by (17.3) and (17.4),

$$\| b_1 \| \leq 2^{3/4} n^{3/4} Y^{3/2} < 2^{3/4} n^{3/4} \left(2^{-7/6} n^{1/6}\right)^{3/2} = 2^{-1} n,$$

we obtain

$$|g(x)| < n.$$

Since $g(x) \equiv 0 \pmod{n}$, we must have $g(x) = 0$. The zeros of $g(T)$ may be determined numerically, and we obtain at most three candidates for x. Each of these may be tried to see if it gives the correct ciphertext. Therefore, Eve can find x.

Note that the above method replaces the problem of finding a solution to the congruence $f(T) \equiv 0 \pmod{n}$ with the exact, non-congruence, equation $g(T) = 0$. Solving a congruence often requires factoring n, but solving exact equations can be done by numerical procedures such as Newton's method.

In exactly the same way, we can find small solutions (if they exist) to a polynomial congruence of degree d, using a lattice of dimension $d + 1$. Of course, d must be small enough that *LLL* will run in a reasonable time. Improvements to this method exist. Coppersmith ([Coppersmith2]) gave an algorithm using higher-dimensional lattices that looks for small solutions x to a monic (that is, the highest-degree coefficient equals 1) polynomial equation $f(T) \equiv 0 \pmod{n}$ of degree d. If $\left|x\right| \leq n^{1/d}$, then the algorithm runs in time polynomial in log n and d.

Example

Let

$$n = 192784105542869748715759425891 7$$

(which happens to be the product of the primes $p = 757285757575769$ and $q = 2545724696579693$

, but Eve does not know this). Alice is sending the message

$$\textit{The answer is **},$$

where ** denotes a two-digit number. Therefore the message is $m = B + x$ where $B = 200805000114192305180009190000$ and $0 \leq x < 100$. Suppose Alice sends the ciphertext $c \equiv (B+x)^3 \equiv 30326308498619648559464058932 \pmod{n}$. Eve forms the polynomial

$$f(T) = (B+T)^3 - c \equiv T^3 + a_2T^2 + a_1T + a_0 \pmod{n},$$

where

$$\begin{aligned} a_2 &= 600241500034257691554002757000 \\ a_1 &= 1123549124004247469362171467964 \\ a_0 &= 58732411444567987695445792761 6. \end{aligned}$$

Note that $a_0 \equiv B^3 - c \pmod{n}$.

Eve uses *LLL* to find a root of $f(T) \pmod{n}$. She lets $Y = 100$ and forms the vectors

$$v_1 = (n,\ 0,\ 0,\ 0), \quad v_2 = (0,\ 100n,\ 0,\ 0), \quad v_3 = \left(0,\ 0,\ 10^4 n,\ 0\right)$$
$$v_4 = \left(a_0,\ 100a_1,\ 10^4{}_{a2},\ 10^6\right).$$

The *LLL* algorithm produces the vector

$$\begin{aligned} &308331465484476402v_1 + 589837092377839611v_2 \\ &\quad +316253828707108264v_3 - 1012071602751202635v_4 \\ &= (246073430665887186108474,\ -577816087453534232385300, \\ &\quad 405848565585194400880000,\ -1012071602751202635000000). \end{aligned}$$

Eve then looks at the polynomial

$$\begin{aligned} g(T) \quad = \quad & -1012071602751202635T^3 + 40584856558519440088T^2 \\ & -5778160874535342323853T + 246073430665887186108474. \end{aligned}$$

The roots of $g(T)$ are computed numerically to be

$$42.000000000, \qquad -0.949612039 \pm 76.079608511i.$$

It is easily checked that $g(42) = 0$, so the plaintext is

```
The answer is 42.
```

Of course, a brute force search through all possibilities for the two-digit number x could have been used to find the answer in this case. However, if n is taken to be a 200-digit number, then Y can have around 33 digits. A brute force search would usually not succeed in this situation.

23.4 NTRU

If the dimension n is large, say $n \geq 100$, the *LLL* algorithm is not effective in finding short vectors. This allows lattices to be used in cryptographic constructions. Several cryptosystems based on lattices have been proposed. One of the most successful current systems is NTRU (rumored to stand for either "Number Theorists aRe Us" or "Number Theorists aRe Useful"). It is a public key system. In the following, we describe the algorithm for transmitting messages using a public key. There is also a related signature scheme, which we won't discuss. Although the initial description of NTRU does not involve lattices, we'll see later that it also has a lattice interpretation.

First, we need some preliminaries. Choose an integer N. We will work with the set of polynomials of degree less than N. Let

$$f = a_{N-1}X^{N-1} + \cdots + a_0 \quad \text{and} \quad g = b_{N-1}X^{N-1} + \cdots + b_0$$

be two such polynomials. Define

$$h = f * g = c_{N-1}X^{N-1} + \cdots + c_0,$$

where

$$c_i = \sum_{j+k \equiv i} a_j b_k.$$

The summation is over all pairs j, k with $j + k \equiv i \pmod{N}$.

For example, let $N = 3$, let $f = X^2 + 7X + 9$, and let $g = 3X^2 + 2X + 5$. Then the coefficient c_1 of X in $f * g$ is

$$a_0b_1 + a_1b_0 + a_2b_2 = 9 \cdot 2 + 7 \cdot 5 + 1 \cdot 3 = 56,$$

and

$$f * g = 46X^2 + 56X + 68.$$

From a slightly more advanced viewpoint, $f * g$ is simply multiplication of polynomials mod $X^N - 1$ (see Exercise 5 and Section 3.11).

NTRU works with certain sets of polynomials with small coefficients, so it is convenient to have a notation for them. Let

$$L(j, k) = \begin{array}{l} \text{the set of polynomials of degree } < N \\ \text{with } j \text{ coefficients equal to } +1 \\ \text{and } k \text{ coefficients equal to } -1. \\ \text{The remaining coefficients are } 0. \end{array}$$

We can now describe the NTRU algorithm. Alice wants to send a message to Bob, so Bob needs to set up his public key. He chooses three integers N, p, q with the requirements that $\gcd(p, q) = 1$ and that p is much smaller than q. Recommended choices are

$$(N, p, q) = (107, 3, 64)$$

for moderate security and

$$(N, p, q) = (503, 3, 256)$$

for very high security. Of course, these parameters will need to be adjusted as attacks improve. Bob then chooses two secret polynomials f and g with small coefficients (we'll say more about how to choose them later). Moreover, f should be invertible mod p and mod q, which means that there exist polynomials F_p and F_q of degree less than N such that

$$F_p * f \equiv 1 \pmod{p}, \qquad F_q * f \equiv 1 \pmod{q}.$$

Bob calculates

$$h \equiv F_q * g \pmod{q}.$$

Bob's public key is

$$(N, p, q, h).$$

His private key is f. Although F_p can be calculated easily from f, he should store (secretly) F_p since he will need it in the decryption process. What about g? Since $g \equiv f * h \pmod{q}$, he is not losing information by not storing it (and he does not need it in decryption).

Alice can now send her message. She represents the message, by some prearranged procedure, as a polynomial m of degree less than N with coefficients of absolute value at most $(p-1)/2$. When $p = 3$, this means that m has coefficients $-1,\ 0,\ 1$. Alice then chooses a small polynomial ϕ ("small" will be made more precise shortly) and computes

$$c \equiv p\phi * h \ + \ m \pmod{q}.$$

She sends the ciphertext c to Bob.

Bob decrypts by first computing

$$a \equiv f * c \pmod{q}$$

with all coefficients of the polynomial a of absolute value at most $q/2$, then (usually) recovering the message as

$$m \equiv F_p * a \pmod{p}.$$

Why should this work? In fact, sometimes it doesn't, but experiments with the parameter choices given below indicate that the probability of decryption errors is less than 5×10^{-5}. But here is why the decryption is usually correct. We have

$$\begin{aligned} a \equiv f * c &\equiv f * (p\phi * h + m) \\ &\equiv f * p\phi * F_q * g + f * m \\ &\equiv p\phi * g + f * m \pmod{q}. \end{aligned}$$

Since ϕ, g, f, m have small coefficients and p is much smaller than q, it is very probable that $p\phi * g + f * m$, before reducing mod q, has coefficients of absolute value less than $q/2$. In this case, we have equality

$$a = p\phi * g + f * m.$$

Then

$$F_p * a = pF_p * \phi * g + F_p * f * m \equiv 0 + 1 * m \equiv m \pmod{p},$$

so the decryption works.

For $(N, p, q) = (107, 3, 64)$, the recommended choices for f, g, ϕ are

$$f \in L(15, 14), \quad g \in L(12, 12), \quad \phi \in L(5, 5)$$

(recall that this means that the coefficients of f are fifteen 1s, fourteen -1s, and the remaining 78 coefficients are 0).

For $(N, p, q) = (503, 3, 256)$, the recommended choices for f, g, ϕ are

$$f \in L(216, 215), \quad g \in L(72, 72), \quad \phi \in L(55, 55).$$

With these choices of parameters, the polynomials f, g, ϕ are small enough that the decryption works with very high probability.

The reason f has a different number of 1s and -1s is so that $f(1) \neq 0$. It can be shown that if $f(1) = 0$, then f cannot be invertible.

Example

Let $(N, p, q) = (5, 3, 16)$ (this choice of N is much too small for any security; we use it only in order to give an explicit example). Take $f = X^4 + X - 1$ and $g = X^3 - X$. Since

$$(X^3 + X^2 - 1) * (X^4 + X - 1) \equiv 1 \pmod{3},$$

we have

$$F_p = X^3 + X^2 - 1.$$

Also,

$$\begin{aligned} F_q &= X^3+X^2-1 \\ h &= -X^4-2X^3+2X+1 \equiv F_q * g \pmod{16}. \end{aligned}$$

Bob's public key is

$$(N, p, q, h) = (5, 3, 16, \ -X^4-2X^3+2X+1).$$

His private key is

$$f = X^4+X-1.$$

Alice takes her message to be $m = X^2 - X + 1$. She chooses $\phi = X - 1$. Then the ciphertext is

$$c \equiv 3\phi * h + m \equiv -3X^4+6X^3+7X^2-4X-5 \pmod{16}.$$

Bob decrypts by first computing

$$a \equiv f * c \equiv 4X^4-2X^3-5X^2+6X-2 \pmod{16},$$

then

$$F_p * a \equiv X^2-X+1 \pmod{3}.$$

Therefore, Bob has obtained the message.

23.4.1 An Attack on NTRU

Let $h = h_{N-1}X^{N-1} + \cdots + h_0$. Form the $N \times N$ matrix

$$H = \begin{pmatrix} h_0 & h_1 & \cdots & h_{N-1} \\ h_{N-1} & h_0 & \cdots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \cdots & h_0 \end{pmatrix}.$$

If we represent $f = f_{N-1}X^{N-1} + \cdots + f_0$ and $g = g_{N-1}X^{N-1} + \cdots + g_0$ by the row vectors

$$\overline{f} = (f_0, \ \ldots, f_{N-1}) \text{ and } \overline{g} = (g_0, \ \ldots, g_{N-1}),$$

then we see that $\overline{f}H \equiv \overline{g} \pmod{q}$.

Let I be the $N \times N$ identity matrix. Form the $2N \times 2N$ matrix

$$M = \begin{pmatrix} I & H \\ 0 & qI \end{pmatrix}.$$

Let L be the lattice generated by the rows of M. Since $g \equiv f * h \pmod{q}$, we can write $g = f * h + qy$ for some polynomial y. Represent y as an N-dimensional row vector $\overline{y}$, so $(\overline{f},\ \overline{y})$ is a $2N$-dimensional row vector. Then

$$(\overline{f},\ \overline{y})\, M = (\overline{f},\ \overline{g}),$$

so $\left(\overline{f}, \overline{g}\right)$ is in the lattice L (see Exercise 3). Since f and g have small coefficients, $\left(\overline{f}, \overline{g}\right)$ is a small vector in the lattice L. Therefore, the secret information for the key can be represented as a short vector in a lattice. An attacker can try to apply a lattice reduction algorithm to find short vectors, and possibly obtain $\left(\overline{f}, \overline{g}\right)$. Once the attacker has found f and g, the system is broken.

To stop lattice attacks, we need to make the lattice have high enough dimension that lattice reduction algorithms are inefficient. This is easily achieved by making N sufficiently large. However, if N is too large, the encryption and decryption algorithms become slow. The suggested values of N were chosen to achieve security while keeping the cryptographic algorithms efficient.

Lattice reduction methods have the best success when the shortest vector is small (more precisely, small when compared to the $2N$th root of the determinant of the $2N$-dimensional lattice). Improvements in the above lattice attack can be obtained by replacing I in the upper left block of M by αI for a suitably chosen real number α. This makes the resulting short vector $\left(\alpha\overline{f}, \overline{g}\right)$ comparatively shorter and thus easier to find. The parameters in NTRU, especially the sizes of f and g, have been chosen so as to limit the effect of these lattice attacks.

So far, the NTRU cryptosystem appears to be strong; however, as with many new cryptosystems, the security is still being studied. If no successful attacks are found, NTRU will have the advantage of providing security comparable to RSA and other public key methods, but with smaller key size and with faster encryption and decryption times.

23.5 Another Lattice-Based Cryptosystem

Another lattice-based public key cryptosystem was developed by Goldreich, Goldwasser, and Halevi in 1997. Let L be a 300-dimensional lattice that is a subset of the points with integral coordinates in 300-dimensional real space $\mathbb{R}^{300}$.

The private key is a “good” basis of L, given by the columns of a 300×300 matrix G, where “good” means that the entries of G are small.

The public key is a “bad basis” of L, given by the columns of a 300×300 matrix $B = G\,U$, where U is a secret 300×300 matrix with integral entries and with determinant 1. The determinant condition implies that the entries of U^{-1} are also integers. “Bad” means that B has many large entries.

A message is a 300-dimensional vector $\vec{m}$ with integral entries, which is encrypted to obtain the ciphertext

$$\vec{c} = B\vec{m} + \vec{e},$$

where $\vec{e}$ is a 300-dimensional vector whose entries are chosen randomly from $\{0,\ \pm 1,\ \pm 2,\ \pm 3\}$.

The decryption is carried out by computing

$$B^{-1}G\lfloor G^{-1}\vec{c}\rceil,$$

where $\lfloor \vec{v}\, \rceil$ for a vector means we round off each entry to the nearest integer (and $.5$ goes whichever way you specify). Why does this decryption work? First, $U = G^{-1}B$, so

$$G^{-1}\vec{c} = G^{-1}B\vec{m} + G^{-1}\vec{e} = U\vec{m} + G^{-1}\vec{e}.$$

Since U and $\vec{m}$ have integral entries, so does the $U\vec{m}$. Since G is good, the entries of $G^{-1}\vec{e}$ tend to be small fractions, so they disappear in the rounding. Therefore, $\lfloor G^{-1}\vec{c}\rceil$ probably equals $U\vec{m}$, so

$$B^{-1}G\lfloor G^{-1}\vec{c}\rceil = B^{-1}GU\vec{m} = B^{-1}GG^{-1}B\vec{m} = \vec{m}.$$

Example

To keep things simple, we'll work in two-dimensional, rather than 300-dimensional, space. Let

$$G = \begin{pmatrix} 5 & 2 \\ 1 & 4 \end{pmatrix} \qquad \text{and} \qquad B = GU = \begin{pmatrix} 5 & 2 \\ 1 & 4 \end{pmatrix}\begin{pmatrix} 17 & 18 \\ 16 & 17 \end{pmatrix} = \begin{pmatrix} 117 & 124 \\ 81 & 86 \end{pmatrix}$$

and

$$\vec{m} = \begin{pmatrix} 59 \\ 37 \end{pmatrix}.$$

Then

$$\vec{c} = \begin{pmatrix} 117 & 124 \\ 81 & 86 \end{pmatrix}\begin{pmatrix} 59 \\ 37 \end{pmatrix} + \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 11492 \\ 7960 \end{pmatrix}.$$

To decrypt, we first compute

$$\lfloor G^{-1}\vec{c}\rceil = \left\lfloor \begin{pmatrix} 2/9 & -1/9 \\ -1/18 & 5/18 \end{pmatrix}\begin{pmatrix} 11492 \\ 7960 \end{pmatrix} \right\rceil = \left\lfloor \begin{pmatrix} 1669.33 \\ 1572.67 \end{pmatrix} \right\rceil = \begin{pmatrix} 1669 \\ 1573 \end{pmatrix}.$$

Therefore, the decryption is

$$B^{-1}G\lfloor G^{-1}\vec{c}\rceil = U^{-1}\lfloor G^{-1}\vec{c}\rceil = \begin{pmatrix} 17 & -18 \\ -16 & 17 \end{pmatrix}\begin{pmatrix} 1669 \\ 1573 \end{pmatrix} = \begin{pmatrix} 59 \\ 37 \end{pmatrix} = \vec{m}.$$

Suppose, instead, that we tried to decrypt by computing $\lfloor B^{-1}\vec{c}\rceil$? In the present example,

$$B^{-1} = \begin{pmatrix} 43/9 & -62/9 \\ -9/2 & 13/2 \end{pmatrix}$$

and

$$B^{-1}\vec{c} = \begin{pmatrix} 43/9 & -62/9 \\ -9/2 & 13/2 \end{pmatrix}\begin{pmatrix} 11492 \\ 7960 \end{pmatrix} = \begin{pmatrix} 212/3 \\ 26 \end{pmatrix}.$$

This rounds off to $(71, 26)$, which is nowhere close to the original message. The problem is that the entries of B^{-1} are much larger than those of G^{-1}, so the small error introduced by $\vec{e}$ is amplified by B^{-1}.

Attacking this system involves the **Closest Vector Problem:** Given a point P in $\mathbf{R}^n$, find the point in L closest to P.

We have $B\vec{m} \in L$, and $\vec{c}$ is close to $B\vec{m}$ since it is moved off the lattice by the small vector $\vec{e}$.

For general lattices, the Closest Vector Problem is very hard. But it seems to be easier if the point is very close to a lattice point, which is the case in this cryptosystem. So the actual level of security is not yet clear.

23.6 Post-Quantum Cryptography?

If a quantum computer is built (see Chapter 25), cryptosystems based on factorization or discrete logs will become less secure. An active area of current research involves designing systems that cannot be broken by a quantum computer. Some of the most promising candidates seem to be lattice-based systems since their security does not depend on the difficulty of computing discrete logs or factoring, and no attack with a quantum computer has been found. Similarly, the McEliece cryptosystem, which is based on error-correcting codes (see Section 24.10) and is similar to the system in Section 23.5, seems to be a possibility.

One of the potential difficulties with using many of these lattice-based systems is the key size: In the system in Section 23.5, the public key requires integer entries. Many of the entries of should be large, so let's say that we use 100 bits to specify each one. This means that the key requires bits, much more than is used in current public key cryptosystems.

For more on this subject, see [Bernstein et al.].

23.7 Exercises

1. Find a reduced basis and a shortest nonzero vector in the lattice generated by the vectors $(58,\ 19),\ \ (168,\ 55)$.

2.
    1. Find a reduced basis for the lattice generated by the vectors $(53,\ 88)$, $(107,\ 205)$.

    2. Find the vector in the lattice of part (a) that is closest to the vector $(151,\ 33)$. (Remark: This is an example of the **closest vector problem**. It is fairly easy to solve when a reduced basis is known, but difficult in general. For cryptosystems based on the closest vector problem, see [Nguyen-Stern].)

3. Let $v_1,\ \ldots,\ v_n$ be linearly independent row vectors in $\mathbf{R}^n$. Form the matrix M whose rows are the vectors v_i. Let $a = (a_1,\ \ldots,\ a_n)$ be a row by $v_1,\ \ldots,\ v_n$, and show that every vector in the lattice can be written in this way.

4. Let $\{v_1,\ v_2\}$ be a basis of a lattice. Let $a,\ b,\ c,\ d$ be integers with $ad - bc = \pm 1$, and let

$$w_1 = av_1 + bv_2, \qquad w_2 = cv_1 + dv_2.$$

    1. Show that

$$v_1 = \pm(dw_1 - bw_2), \qquad v_2 = \pm(-cw_1 + aw_2).$$

    2. Show that $\{w_1,\ w_2\}$ is also a basis of the lattice.

5. Let N be a positive integer.

    1. Show that if $j + k \equiv i \pmod N$, then $X^{j+k} - X^i$ is a multiple of $X^N - 1$.

    2. Let $0 \le i < N$. Let $a_0,\ \ldots,\ a_{N-1},\ b_0,\ \ldots,\ b_{N-1}$ be integers and let

$$c_i = \underset{j+k\equiv i}{} a_j b_k,$$

where the sum is over pairs $j,\ k$ with $j + k \equiv i \pmod N$. Show that

$$c_i X^i - \underset{j+k\equiv i}{} a_j b_k X^{j+k}$$

is a multiple of $X^N - 1$.

3. Let f and g be polynomials of degree less than N. Let fg be the usual product of f and g and let $f * g$ be defined as in Section 23.4. Show that $fg - f * g$ is a multiple of $X^N - 1$.

6. Let N and p be positive integers. Suppose that there is a polynomial $F(X)$ such that $f(X) * F(X) \equiv 1 \pmod{p}$. Show that $f(1) \not\equiv 0 \bmod p$. (Hint: Use Exercise 5(c).)

7.
    1. In the NTRU cryptosystem, suppose we ignore q and let $c = p\phi * h + m$. Show how an attacker can obtain the message quickly.
    2. In the NTRU cryptosystem, suppose q is a multiple of p. Show how an attacker can obtain the message quickly.