Lectures on the NTRU encryption algorithm and digital signature scheme: Grenoble, June 2002

J. Pipher

Brown University, Providence RI 02912

1 Lecture 1

1.1 Integer lattices

Lattices have been studied by cryptographers for quite some time, in both the field of cryptanalysis (see for example [16–18]) and as a source of hard problems on which to build encryption schemes (see [1, 8, 9]). In this lecture, we describe the NTRU encryption algorithm, and the lattice problems on which this is based. We begin with some definitions and a brief overview of lattices.

If $\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n$ are n independent vectors in $\mathbb{R}^m$, $n \leq m$, then the integer lattice with these vectors as basis is the set $\mathcal{L} = \{\sum_1^n x_i \mathbf{a}_i : x_i \in \mathbb{Z}\}$. A lattice is often represented as matrix A whose rows are the basis vectors $\mathbf{a}_1, ..., \mathbf{a}_n$. The elements of the lattice are simply the vectors of the form $\mathbf{v}^T A$, which denotes the usual matrix multiplication. We will specialize for now to the situation when the rank of the lattice and the dimension are the same ($n = m$).

The determinant of a lattice $\det(\mathcal{L})$ is the volume of the fundamental parallelepiped spanned by the basis vectors. By the Gram-Schmidt process, one can obtain a basis for the vector space generated by $\mathcal{L}$, and the $\det(\mathcal{L})$ will just be the product of these orthogonal vectors. Note that these vectors are not a basis for $\mathcal{L}$ as a lattice, since $\mathcal{L}$ will not usually possess an orthogonal basis.

The two fundamental problems in the theory of integer lattices are the shortest vector problem (SVP), and the more general closest vector problem (CVP). Roughly speaking, both of these problems are concerned with finding the "most efficient" basis of the lattice - a basis consisting of vectors which are as short and as orthogonal as possible. Specifically, if $\mathbf{v}$ is a vector in the lattice, let $||\mathbf{v}||$ denote a norm on $\mathbf{v}$, typically the sup norm, $\max \mathbf{v}_i$, or the Euclidean norm $\sqrt{\sum \mathbf{v}_i^2}$. Given a basis A for a lattice $\mathcal{L}$, the shortest vector problem is that of finding a nonzero vector in $\mathcal{L}$ with minimum norm. Given an arbitrary (target) vector $\mathbf{v}$ in

$\mathbb{Z}^n$, the CVP is the problem of finding the lattice point $\mathbf{x}^T A$ closest in norm to $\mathbf{v}$, i.e., of minimizing $||A\mathbf{x} - \mathbf{v}||$. No polynomial time algorithms exist for solving either of these problems, although they have been well studied both experimentally and theoretically. There are algorithms which find approximations to the shortest vector - we'll say more on that in a moment. First, let's investigate a simpler question - that of predicting the length of the shortest vector.

The relationship between the minimum distance between lattice points and geometric properties of the lattices is an example of a question in the geometry of numbers, a field essentially formed when Minkowski proved the following theorem.

Theorem 1. *If $\mathcal{L}$ is a lattice in $\mathbb{R}^n$ (of rank n), then any convex set K of area greater than $2^n det(\mathcal{L})$ which is symmetric about the origin must contain a nonzero lattice point.*

This result gives an upper bound for the length of the "expected shortest vector", λ_e in a random lattice. Suppose that $K = B(r)$ be the ball centered at the origin of radius r, and choose r so that $vol(B(r))$ is exactly $2^n \det(\mathcal{L})$. Then $B(r)$ should contain a nonzero lattice point - and r should be an upper bound for the minimum length of the shortest vector in the Euclidean norm. In $\mathbb{R}^n$, the volume of $B(r)$ is $\frac{\pi^{n/2} r^n}{\Gamma(n/2+1)}$, and approximating $\Gamma(x+1)$ by $\sqrt{2\pi x}(x/e)^x$, we find that $\lambda_e \leq \sqrt{n/\pi e}\det(\mathcal{L})^{1/n}$. A general principle (the so-called "Gaussian heuristic") estimates the number of lattice points in the ball by the volume of the set divided by the determinant of the lattice. This gives an intuitive, and commonly used, estimate of $\det(\mathcal{L})^{1/n}\sqrt{n/2\pi e}$ for λ_e. We will see that the ratio of this value to the actual length of the shortest vector in a lattice will be an important constant in evaluating the security of NTRU lattices.

The closest vector problem is a hard problem, in the sense of complexity theory - the CVP is known to be NP-complete, in any norm. While the SVP is believed to be an NP-hard, this is an open problem. There are a number of indications that SVP is hard - see the recent work of Micciancio [6], for example. The book "Complexity of Lattice Problems" by Micciancio and Goldwasser is a good reference for these issues in computational complexity, which we'll not discuss further here. For us, the important fact is that there is no polynomial time algorithm which finds the shortest vector. There is however a well known algorithm due to Lenstra, Lenstra and Lovasz [5], which finds an approximately short vector - guaranteed to be within a factor $(2/\sqrt{3})^n$ of the actual shortest - in polynomial time. LLL produces a "reduced" basis of a lattice,

producing an approximately short basis vector as a result. The security of the NTRU public key cryptosystem will ultimately rest on the inability of LLL (or any of its more modern variants) to produce particularly good short vectors within a reasonable amount of time. While LLL often works better in practice (finding shorter vectors than it is guaranteed to produce), it appears that the running time of the LLL algorithm grows exponentially with the dimension. We'll return to the details after describing the NTRU lattices. Further details specific to NTRU lattices can be found in the appendix.

1.2 Polynomial rings and the basic NTRU operations

The basic NTRU operations take place in the ring of convolution polynomials $R = \mathbb{Z}[X]/(X^N - 1)$, where N is prime. An element $F \in R$ will be written as a polynomial or a vector,

$$F = \sum_{i=0}^{N-1} f_i x^i = [F_0, F_1, \ldots, F_{N-1}].$$

We write $*$ to denote multiplication in R. This $*$ multiplication is given explicitly as a cyclic convolution product, $F * G = H$, where

$$H_k = \sum_{i=0}^{k} F_i G_{k-i} + \sum_{i=k+1}^{N-1} F_i G_{N+k-i} = \sum_{i+j=k mod N} F_i G_j.$$

When we do a multiplication modulo q, we mean to reduce the coefficients modulo q. The parameter q need not be prime.

The obvious way to measure the size of an element $F \in \mathbb{Z}[X]/(X^N-1)$ is by the Euclidean norm $(\sum F_i)^{1/2}$ of it vector of coefficients. However, when working with the ring R and its sublattices, it is better to work with the centered norm, which is defined in the following way. Let $m_F = (1/N)\sum F_i$ denote the average of the coefficients of the polynomial $F(X) \in R$. Then the centered norm of F, also denoted $\|F\|$ is defined by

$$\|F\|^2 = \sum_{i=0}^{N-1} (F_i - m_F)^2 = \sum_{i=0}^{N-1} F_i^2 - \frac{1}{N}\left(\sum_{i=0}^{N-1} F_i\right)^2. \qquad (1)$$

For randomly chosen polynomials F and G, the norm is quasi-multiplicative,

$$\|F * G\| \approx \|F\| \cdot \|G\|.$$

This quasi-multiplicative property can be explained via a computation related to Khintchine's inequality: Suppose that $\{f_i\}$ is a random N length sequence with $\text{Prob}(f_i = 1) = d$, $\text{Prob}(f_i = -1) = d$ and $\text{Prob}(f_i = 0) = N - 2d$, and $\{g_i\}$ is any sequence with l^2 norm $\|g\| = \sqrt{\sum g_i^2}$. Then

$$\text{Prob}(|\sum_{i=1}^{N} f_i g_i| > \frac{\lambda\sqrt{2d}\|g\|}{\sqrt{N}}) \leq (\frac{2d}{N})^N e^{-d\lambda^2/N}.$$

When considering n-tuples of elements of R, there is no reason that they should be centered around the same value. In general we define the centered norm of an n-tuple $(a, b, \ldots, c)$ with $a, b, \ldots, c \in R$ by the formula

$$\|(a, b, \ldots, c)\|^2 = \|a\|^2 + \|b\|^2 + \cdots + \|c\|^2. \tag{2}$$

It will also be necessary to compute inverses (or almost-inverses) within the ring $R_q = \mathbb{Z}/q\mathbb{Z}[X]/(X^N - 1)$, where q is a prime, or a power of a prime. If F is a polynomial in R_q such that $F(1)$ has a factor in common with q, then it cannot be invertible in R_q. In particular, $F(1)$ should not equal 0. Thus not every polynomial in this ring is invertible, but it can be shown that most polynomials, even among those which satisfy, say $F(1) = 1$, have inverses. To determine how many elements of R_q are invertible, where q is a power of a prime p, it suffices to count the number of invertible elements in R_p. In R_p, the polynomial $X^N - 1$ factors into $(N-1)/n$ irreducible polynomials of degree n, where n is the smallest integer such that $p^n = 1 \pmod{N}$. Thus the number of invertible elements in R_p equals $(p-1)(p^n-1)^{(N-1)/n)}$ and the probability that a randomly chosen element of this ring is invertible is this number divided by p^N, or

$$(1 - \frac{1}{p})(1 - \frac{1}{p^n})^{(N-1)/n}.$$

If it is also required that $F(1) = 1$, a computation shows that probability is still is large as

$$(1 - \frac{1}{p^n})^{(N-1)/n}.$$

Units modulo p can be lifted to units modulo powers of p, and so finding inverses in R_q can be reduced to using the Euclidean algorithm in R_p. We shall want as many invertible elements as possible, and with respect to two different primes - optimal choices make n as large as possible. In addition, there are algorithms which provide faster ways of computing inverses.

1.3 NTRU Encryption algorithm

Public parameters . A choice of N determines the polynomial ring $\mathbb{Z}[X]/(X^N - 1)$. Two moduli p and q are selected so that $\gcd(p, q) = 1$ - here q will typically be a power of 2, and p will be very small. One example is $(N, p, q) = (251, 3, 128)$. Additional public parameters are numbers, which, for reasons that will be apparent momentarily, we'll denote d_f, d_g, d_m and d_r. These specify the space of allowable private keys f and g, the allowable messages, and the form of the random polynomial r used in encryption.

Key creation Choose random "small" polynomials f and g, where f has d_f 1's and $d_f - 1$ (-1)'s, and the rest zeroes, while g will be similar, but have the same number (d_g) of 1's and (-1)'s. By construction, $f(1) = 1$, but, in addition, f will need to be invertible in $\mathbb{Z}[X]/(X^N - 1)$ modulo p and q. (As mentioned above, this occurs with high probability - when it fails, a new random f is generated.) The inverse $f_q^{-1} \bmod q$ is computed, and the polynomial

$$h = f_q^{-1} g \pmod{q}$$

is published. This h is the public key, while both f and g are private.

Encryption Choose a random polynomial r with d_r 1's, d_r (-1)'s and the rest zeroes. Let m be the (trinary) message, with d_m 1's and (-1)'s. Compute the ciphertext

$$e = m + pr * h \pmod{q}$$

. Reduction mod q here means reduction of the coefficients into the interval $(-q/2, q/2]$. (This is the natural choice since these polynomials are have coefficients centered around zero.)

Decryption Multiply e by the private key f to obtain

$$f * e = f * m + pr * f * h \pmod{q} = f * m + pr * g \pmod{q},$$

where this last equality uses the definition of h. With an appropriate choice of parameters which determine the sizes of the private keys, messages, and blinding polynomial r, it turns out that the coefficients of $f * m + pr * g$ will naturally lie in the interval $(-q/2, q/2]$. That is, the very last reduction mod q above was superfluous. Hence, having obtained exactly $f * m + pr * g$, we can now reduce this mod p to recover $f * m$, and m is recovered after multiplying by the inverse f_p^{-1} of f mod p. Note that m is a mod p polynomial, and is then recovered exactly.

The basic idea behind the decryption step is the following observation about convolution multiplication of small polynomials. A coefficient of, say, $f * m$ is a sum of products of the form $f_i m_j$, each of which take on values 0, 1 and -1 with some probability. Let's think of a coefficient as a random variable, and consider what sort of distribution it is likely to possess. While it is reasonable to suppose that this random variable is normally distributed around zero, it turns out that the hypergeometric distribution (same mean, smaller standard deviation) is more accurate. Thus the coefficients of products of small zero-centered polynomials like $f * m$ stay very tightly clustered around zero. (Of course, f is not exactly centered here.) Therefore it is easy to determine (experimentally) choices of parameters for which decryption happens with overwhelming probability. For example, in order for the decryption process to work, it is necessary that

$$|f * m + pr * g|_\infty < q.$$

We have found that this will virtually always be true if we choose parameters so that

$$|f * m|_\infty \leq q/4 \qquad \text{and} \qquad |pr * g|_\infty \leq q/4,$$

When decryption fails, there are two possible reasons. If not all the coefficients of $b = f * m + pr * g$ lie in $(q/2, q/2]$, but yet satisfy

$$\max b_i - \min b_i \leq q$$

, then we say that a wrap failure has occured. A wrap failure can be fixed by shifting all the coeffiecients. If however the spread between the max and the min of the coefficients is larger than q, then a "gap" failure has occurred. Again, a judicious choice of parameters will make such failures extremely unlikely.

1.4 Some basic NTRU security issues

There are number of different types of security issues: elementary (why N should be prime), standard (meet-in-the-middle attacks), sophisticated (lattice reduction attacks) and implementational (chosen ciphertext attacks, padding and hashing issues). Let's talk first about the underlying hard problem in NTRU encryption.

NTRU lattices The hard problem underlying NTRU is the CVP problem in some special (convolution modular) lattices. The private key pair (f, g) will be a fairly short vector in the integer lattice $\mathcal{L}_h$ represented by the matrix

$$\left(\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{N-1} \\ 0 & 1 & \cdots & 0 & h_{N-1} & h_0 & \cdots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & h_1 & h_2 & \cdots & h_0 \\ \hline 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array}\right)$$

which we shall abbreviate

$$M = \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}.$$

If k is the integer polynomial such that $f * h = g + qk$, then the vector (f, k) is the linear combination of the rows of this matrix which produces the private key pair (f, g). The h-vectors above have coefficients which are fairly uniformly distributed in a q-interval. (Inverses mod q of small polynomials can't be particularly distinguished from arbitrary mod q polynomials in this ring.) This lattice has dimension $2N$ (and determinant q^N) whereas the public key has length $N \log q$ (h is an N-length vector with integer entries mod q). The ability to keep the key length relatively small compared to the dimension of the lattice in which the private key is a short vector distinguishes NTRU from other lattice based cryptosystems, such as the knapsack or GGH systems ([8]). Thus the key length is of practical size, while the dimension of the lattice is outside the range of existing lattice reduction technology. In practice, the parameters of the NTRU scheme will be chosen to balance a number of competing considerations: minimizing decryption failure, eliminating the possibility of brute force searches, maintaining speed and efficiency, and requiring an impossibly large estimated (breaking) time for the LLL algorithm to return a short vector.

Remark 1. This basic scheme, and the NTRU lattice above, was presented at Crypto '96, where preprints were distributed. At that time, we

(J. Hoffstein, J. Silverman and myself) had not completed the details of the analysis of the predicted breaking times of LLL, and our sample parameters were preliminary. Shortly after this, D. Coppersmith and A. Shamir ([2]) made two important observations. First, they pointed out that the lattice which optimizes the ability of LLL to return short vectors will in fact be the one represented by the matrix

$$M = \begin{pmatrix} \alpha\, h \\ 0\ q \end{pmatrix},$$

where α denotes the $N \times N$ identity matrix multiplied by a certain constant α. (See the discussion following the next remark for a definition of α. They also pointed out that an attacker need not find the exact private key pair - indeed, any short vector (f', g') in the above lattice can be used to decrypt. These two points were made in their paper [], presented at Eurocrypt '97. The paper also expressed the opinion that the security of NTRUEncryptwas fragile and temporary. However, subsequent experiments using LLL on these lattices reveal that efficient and quite secure parameter choices are possible, and that the security of the algorithm is more robust than it may have initially appeared.

Remark 2. The Hermite Normal Form of a matrix (lattice) is upper triangular, with all entries positive, and the diagonal entries the largest in that row. Every integer lattice can be put into Hermite Normal form, and this HNF can be efficiently computed. We note that the natural lattice for NTRU is already in HNF.

Let $\mathcal{L}_h$ be the lattice generated by the rows of this matrix. The determinant of $\mathcal{L}$ is $q^N \alpha^N$. (We shall drop the subscript h when the context is clear. Since the public key is $h = g * f^{-1}$, the lattice $\mathcal{L}$ will contain the vector $\tau = (\alpha f, g)$, by which we mean the $2N$ vector consisting of the N coefficients of f multiplied by α, followed by the N coefficients of g. By the Gaussian heuristic, the expected size of the smallest vector in a random lattice of dimension n and determinant D is approximately $D^{1/n}\sqrt{\frac{n}{2\pi e}}$ In our case, $n = 2N$ and $D = q^N \alpha^N$, so the expected smallest length is larger (but not much larger) than

$$s = \sqrt{\frac{N\alpha q}{\pi e}}.$$

An implementation of a lattice reduction algorithm will have the best chance of locating $\tau = (f, g)$, or another vector whose length is close to

τ, if the attacker chooses α to maximize the ratio $s/|\tau|_2$. Squaring this ratio, we see that an attacker should choose α so as to maximize

$$\frac{\alpha}{\alpha^2 |f|_2^2 + |g|_2^2} = \left(\alpha |f|_2^2 + \alpha^{-1} |g|_2^2\right)^{-1}.$$

This is done by choosing $\alpha = |g|_2 / |f|_2$. (Note that $|g|_2$ and $|f|_2$ are both public quantities.)

When α is chosen in this way, we define a constant c_h by setting $|\tau|_2 = c_h s$. Thus c_h is the ratio of the length of the target vector to the length of the expected shortest vector. The smaller the value of c_h, the easier it will be to find the target vector. Substituting in above, we obtain

$$c_h = \sqrt{\frac{2\pi e |f|_2 |g|_2}{Nq}}.$$

For a given pair (f, g) used to set up the cryptosystem, c_h may be viewed as a measure of how far the associated lattice departs from a random lattice. If c_h is close to 1, then L will resemble a random lattice and lattice reduction methods will have a hard time finding a short vector in general, and finding τ in particular. As c_h decreases, lattice reduction algorithms will have an easier time finding τ. Based on extensive experimental evidence, the time required appears to be (at least) exponential in N, with a constant in the exponent proportional to c_h.

We note that decryption of the message is equivalent to finding the vector $(pr, pr * h \bmod q$ which is a vector in the lattice close to the (non-lattice) vector $(0, m)$. Thus decryption solves a CVP problem for the the NTRU lattice.

The NTRU lattices are special cases of convolution modular lattices ones, consisting of $N \times N$ blocks of circulant matrices. Such convolution modular lattices have a rotational invariance property, since if $(u, v) \in \mathcal{L}$, then

$$(X^i * u, X^i * v) \in \mathcal{L} \quad \text{for all } 0 \le i < N.$$

Notice that each of the rotations $(X^i * u, X^i * v)$ has the same (centered) norm as (u, v).

Here are a few final general remarks about the relationship between the parameters:

1. The security parameters N and q are related by

$$q = O(N).$$

In practice, we generally take $\frac{1}{3}N \le q \le \frac{2}{3}N$.

2. The small polynomials have coefficients of size $O(1)$, that is, coefficients whcih are bounded independently of N. Therefore the (centered) norm of a small polynomial $a(X)$ satisfies

$$\|a\| = O(\sqrt{N}).$$

3. A general convolution modular lattice $\mathcal{L}$ has dimension $2N$ and determinant q^N, so its probable shortest vector and closest vectors have size approximately

$$\lambda_e(\mathcal{L}) = \sqrt{Nq/\pi e} = O(N). \tag{3}$$

   Notice that $\mathcal{L}$ contains N linearly independent vectors of length $q = O(N)$, namely the rightmost N columns of its matrix. Small linear combinations of these "q-vectors" are the only obvious vectors of length $O(N)$ in this lattice.
4. If $(u, v) \in \mathcal{L}$, then the vector obtained by reducing the coordinates of u and v modulo q is in $\mathcal{L}$. Thus this lattice contains a large number of vectors of length $O(q\sqrt{N}) = O(N^{3/2})$.
5. An NTRU lattice contains not only the short vector (f, g), but all of its (short) rotations $(X^i * f, X^i * g)$, which have length $O(\sqrt{N})$. Based on the Gaussian heuristic, these secret short vectors are probably $O(\sqrt{N})$ smaller than any vector not in the subspace $R * (f, g)$ that they span.

At this point, we briefly discuss two additional basic security issues which dictate the choice of parameters.

Meet-in-the-middle attacks To cut down the search space for the private key, one can split f in half, say $f = f_1 + f_2$, and then one matches $f_1 * h$ against $-f_2 * h$, looking for the pair (f_1, f_2) so that the corresponding coefficients have approximately the same value. Hence in order to obtain a security level of (say) 2^{80}, one must choose d_f and d_g so that the space of allowable pairs contains around 2^{160} elements. Hence the security level from a brute force search aided by this trick is given by

$$\begin{pmatrix} \text{Key} \\ \text{Security} \end{pmatrix} = \frac{1}{d_g!}\sqrt{\frac{N!}{(N-2d_g)!}}$$
$$\begin{pmatrix} \text{Message} \\ \text{Security} \end{pmatrix} = \frac{1}{d!}\sqrt{\frac{N!}{(N-2d)!}},$$

assuming that the d_g is the smaller of d_f, d_g.

Composite N, and the discrete Fourier transform A variety of suggested parameters exist in the literature for NTRU encryption - always using prime choices for N, the lengths of the key vectors. However, Craig Gentry, in [ ], was the first to publish the observation that if N is composite, the dimension of the lattice that might be used to attempt recovery the keys may be smaller than 2N, thereby reducing security. Gentry's idea involved a factorization of the the polynomial rings $\mathbb{Z}[X]/(X^N-1)$ - we'll take a different approach here.

It is tempting to choose N composite in order to use devices like the Fast Fourier Transform (FFT) to reduce the computation required to compute the convolution products. However, it is exactly the computation in FFT which shows how to reduce the dimension of the lattice. The discrete Fourier transform (DFT) of an N length vector $(f_0, f_1, ..., f_{N-1})$ is another N vector, denoted $\mathcal{F}_N(f)$, whose kth coefficient is defined by

$$\mathcal{F}_N(f)[k] = \sum_{i=0}^{N-1} f_i W^{-ik}$$

where W is a primitive Nth root of unity, i.e., and $W^N = 1$ (and hence $1+W+W^2+...+W^{N-1} = 0$). The objects in the ring $\mathbb{Z}[X]/(X^N-1)$ may thus be identified with Fourier transforms of integer vectors. The main property that we'll use is the interaction between the Fourier transform, and the convolution, namely, that

$$\mathcal{F}_N(a * b)[k] = \mathcal{F}_N(a)[k]\mathcal{F}_N(b)[k].$$

The FFT reduces the number of computations required in computing the FT by regrouping the terms efficiently. So if N is divisible by 2, say, the FFT will cut the number of computations in half. It works as follows. Suppose that $N = 2M$, and for any N vector f, let f_{even} be the vector of length M whose kth coefficient, call it $f_{even}(k)$, is equal to $f_k + f_{k+M}$, for $k = 0, 1, ..., M-1$. Let $W^N = 1$ and hence $W^M = -1$ and W^2 is a primitive Mth root of unity. Then it is not hard to see that for all $i = 2k$,

$$\mathcal{F}_N(f)[2k] = \mathcal{F}_M(f_{even}[k].$$

Applying this to the convolution product $h * f$, it follows that

$$\mathcal{F}_N(f * h)[2k] = \mathcal{F}_N(f)[2k]\mathcal{F}_N(h)[2k] = \mathcal{F}_M(f_{even})[k]\mathcal{F}_M(h_{even})[k] = \mathcal{F}_M(f_{even} * h_{even})[k].$$

In particular, the vector (f_{even}, g_{even}) will be a relatively short vector in an N dimensional lattice - the convolution modular lattice represented by

$$M = \begin{pmatrix} \alpha\, h_{even} \\ 0 \quad q \end{pmatrix}.$$

For $i = 2k + 1$, there is a similar formula. Since

$$\mathcal{F}_N(f)[2k+1] = \sum_0^{N-1} f_i W^{-i} W^{-2ik}$$

the same computation as in the even case shows that

$$\mathcal{F}_N(f)[2k+1] = \mathcal{F}_M(f_{odd})[k],$$

where

$$f_{odd}(i) = f_i W^{-i} + f_{i+M} W^{i+M} = f_i W^{-i} - f_{i+M} W^{-i}.$$

Using this identity in the convolution product, we see that

$$f_{odd} * h_{odd} = g_{odd}.$$

Cancelling the W^{-i} from both sides, we find that the pair (f_o, g_o), where $f_o(k) = f_k - f_{k+M}$, is a short vector in the N dimensional convolution lattice generated by h_o. Thus the 2N dimensional lattice reduction problem is replaced by two N dimensional lattices, presumably recovering both $f_k + f_{k+M}$ and $f_k - f_{k+M}$.

2 Lecture 2

A digital signature scheme should have the following properties: there is an algorithm to produce a signature for a document, there is a means to verify valid signatures, and forgery should be impossible. Often the requirement that valid signatures should always be accepted is relaxed in favor of eliminating the possibility of forgery.

In this lecture we describe the digital signature scheme NTRUSign which is based on the CVP for NTRU modular lattices. Most of the following introduction to NTRUSign is taken from articles and material available on the NTRU web site www.ntru.com, whose authors include J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. Silverman, and W. Whyte.

The core idea of NTRUSign is as follows. The Signer's private key is a short generating basis for an NTRU lattice and his public key is a much longer generating basis for the same lattice. The signature on a digital document is a vector in the lattice with two important properties:

– The signature is attached to the digital document being signed.
– The signature demonstrates an ability to solve a general closest vector problem in the lattice.

The way in which NTRUSIGN achieves these two properties may be briefly summarized as follows:

Key Generation The private key includes a short $2N$-dimensional vector denoted (f, g). The public key is the large N-dimensional vector h that specifies the NTRU lattice L_h^{NT}, that is, h is generated from f and g by the usual NTRU convolution congruence $h \equiv f^{-1} * g \pmod{q}$. The private key also includes a complementary short vector (F, G) that is chosen so that (f, g) and (F, G) generate the full NTRU lattice L_h^{NT}.

Signing The digital document to be signed is hashed to create a random vector (m_1, m_2) modulo q. The signer uses the (secret) short generating vectors to find a lattice vector (s, t) that is close to (m_1, m_2).

Verification The verifier uses the public key h to verify that (s, t) is indeed in the lattice L_h^{NT} and he verifies that (s, t) is appropriately close to (m_1, m_2).

Remark 3. An earlier digital signature scheme called NSS, also based on NTRU lattices, was presented at Eurocrypt 2001 [11]. A number of cryptographers found weaknesses in NSS due to the incomplete linkage between an NSS signature and the underlying hard lattice problem. The private key pair (f, g) which generates h, together with all pairs of rotations of f and of g, yields only N basis vectors for the 2N dimensional NTRU lattice. The scheme NSS attempted to set up a CVP using only these basis vectors, imposing a variety of additional conditions on valid signatures in order to maintain the claim of 2N dimensional lattice security. This weaker solution to the CVP problem for NTRU left the door open to a variety of attacks: a simple forgery of signatures noticed independently by Gentry, Jonsson, Stern, and a more significant inherent difficulty involving the leakage of information in a short transcript of signatures, developed by Gentry and Szydlo. The difficulties created by using only half the basis vectors are resolved in NTRUSIGN, since there is a direct and straightforward linkage between NTRUSIGN signatures and the (approximate) closest vector problem in the underlying NTRU lattice.

Remark 4. The principle upon which NTRUSIGN is based is very simple. The signer has private knowledge of a short basis of vectors in the NTRU

lattice. Given an arbitrary point in space arising from a message digest, the signer uses this knowledge to find a point in the NTRU lattice close to the message point. He then exhibits this approximate solution to the closest vector problem (CVP) as his signature. This basic idea was already proposed by Goldreich, Goldwasser and Halvei in [8].

The fundamental advance of the NTRUSign algorithm is the use of NTRU lattices for CVP-based signatures. The cyclical nature of the NTRU lattices allows the public key to be specified by just one or two vectors, and it is this property that allows secure instances of encryption or signing with practical key sizes. Thus for lattices of dimension n, the GGH proposal [8] requires keys of size $O(n^2)$ bits, while NTRU uses keys of size $O(n \log n)$ bits. At a practical level, this means that a secure version of GGH requires keys with between 10^5 and 10^6 bits (see [16]), while NTRUEncrypt and NTRUSign achieve RSA 1024 bit security with keys of under 2000 bits.

It should be noted that the use of NTRU lattices for CVP-based signatures is not completely straightforward. For the GGH scheme, the signer is free to choose any basis of short vectors for the private key. For an NTRU lattice, the first short vector (f, g) and the public parameters N and q completely determine the lattice $\mathcal{L}$ - so the signer only has a short basis for half of the lattice and must use the known short vector (f, g) to find a complementary short vector (F, G) that, together with (f, g), generates $\mathcal{L}_h$. The efficient construction of an appropriate (F, G) is a nontrivial task.

We will see that the ability to forge signatures will imply the ability to solve an approximate CVP in high dimensions for the NTRU lattices. However, the security of a digital signature scheme, as compared to an encryption scheme, has an extra component: the potential leakage of information in a transcript of signatures built from the same private key. The requirement of a signature scheme that no information can be obtained from a transcript of any length can be relaxed for practical use. In this case, one needs an estimate of how often a new key should be generated to achieve a desired security level. Our analysis shows that the most efficient known method of transcript attack is ineffective on transcipts of 10^8 valid signatures. We benefited a great deal from comments on these topics from C. Gentry, J.Lahtonenen, A. Renvall and M. Szydlo. In particular, Gentry and Szydlo developed the use of higher order moments and their asymptotics in analyzing transcripts, and showed how lattices can be used to aid convergence in certain situations.

2.1 Key generation, signing, verification

An NTRUSign public/private key pair is created exactly as in the public key scheme /NTRUEncrypt: the key creator chooses (f, g) and forms $h \equiv f^{-1} * g \pmod{q}$. However, as part of his private key, the key creator also computes two additional polynomials F and G satisfying

$$f * G - g * F = q, \quad \text{and} \quad \|F\|, \|G\| = O(N).$$

It will turn out that the rotations of (f, g) and (F, G) then form a basis for $\mathcal{L}_h$.

Let's assume for the moment that the signer has such an (F, G) pair at his disposal.

Signing To sign a digital document D, the signer first hashes D to produce a message digest $m = (m_1, m_2)$ composed of two random mod q polynomials m_1 and m_2.

The signature on D is a vector $(s, t) \in \mathcal{L}_h$ that is very close to m. The signer finds (s, t) by expressing (m_1, m_2) as a $\mathbb{Q}$-linear combination of his short basis vectors and then rounding the coefficients to the nearest integer. This standard method of approximately solving a CVP using a "good basis" of a lattice was already suggested for use in cryptography by [8].

Remark 5. The hashing of the document is not only done to reduce the size, there are security issues as well. For example, a signature on a message will also be a valid signature on any other message close to the given one. Thus, to prevent fraud, a small change in a message should produce a large or random change in its hashed image. The security issues related to hashing of the documents will be discussed at the end of this lecture.

Algorithmically, this procedure for the NTRU lattice can be described as follows:

- Compute polynomials $a, b, A, B \in \mathbb{Z}[X]/(X^N - 1)$ by the formulas

$$\begin{aligned} G * m_1 - F * m_2 &= A + q * B, \\ -g * m_1 + f * m_2 &= a + q * b, \end{aligned} \tag{4}$$

  where a and A are chosen to have coefficients between $-q/2$ and $q/2$.
- Compute polynomials s and t as

$$\begin{aligned} s &\equiv f * B + F * b \pmod{q}, \\ t &\equiv g * B + G * b \pmod{q}. \end{aligned} \tag{5}$$

The polynomial s is the signature on the digital document D for the public key h.

Remark 6. In practice, only b and B will be needed to create the signature, and only s is needed to form the signature. We also observe that (s, t) is in the NTRU lattice $\mathcal{L}_h$, since we can write

$$(s, t) = B * (f, g) + b * (F, G) \pmod{q}.$$

Remark 7. For a polynomial $P(X) \in \mathbb{Q}[X]$ with rational coefficients, we use the notation $\lfloor P \rceil$ to denote the polynomial obtained by rounding each coefficient of P to the nearest integer.

Then the full signing process may be summarized by the following matrix equation, which shows that we are using our short basis $\{(f, g), (F, G)\}$ in the standard way to find approximate solutions to CVP:

$$\begin{aligned} \begin{pmatrix} s & t \end{pmatrix} = \begin{pmatrix} B & b \end{pmatrix} \begin{pmatrix} f & g \\ F & G \end{pmatrix} &= \left\lfloor \begin{pmatrix} m_1 & m_2 \end{pmatrix} \begin{pmatrix} G/q & -g/q \\ -F/q & f/q \end{pmatrix} \right\rceil \begin{pmatrix} f & g \\ F & G \end{pmatrix} \\ &= \left\lfloor \begin{pmatrix} m_1 & m_2 \end{pmatrix} \begin{pmatrix} f & g \\ F & G \end{pmatrix}^{-1} \right\rceil \begin{pmatrix} f & g \\ F & G \end{pmatrix} \end{aligned} \tag{6}$$

Verification Let s be a putative NTRUSign signature for the message digest $m = (m_1, m_2)$ and public key h. The signature will be valid if it demonstrates that the signer knows a lattice point in L_h^{NT} that is sufficiently close to the message digest vector m. Verification thus consists of the following two steps:

- Compute the polynomial

  $$t \equiv h * s \pmod{q}.$$

  (Note that by definition, (s, t) is a point in the lattice $\mathcal{L}_h$.)
- Compute the (centered) distance from (s, t) to (m_1, m_2) and verify that it is smaller than a prespecified value `NormBound`.
  In other words, check that

  $$\big\|(s - m_1, t - m_2)\big\| \leq \texttt{NormBound}. \tag{7}$$

A valid signature demonstrates that the signer knows a lattice point that is within `NormBound` of the message digest vector m. Clearly the smaller that `NormBound` is set, the more difficult it will be for a forger, without knowledge of the private key, to solve this problem. It is thus

important to analyze how small we can set the bound `NormBound`, while still allowing valid signatures to be efficiently generated by the signer.

From (4) and (5) (or using (6)), we can calculate

$$(m_1, m_2) - (s, t) = \begin{pmatrix} A/q & a/q \end{pmatrix} \begin{pmatrix} f & g \\ F & G \end{pmatrix}.$$

We recall that the coefficients of a and A are between $-q/2$ and $q/2$, and hence

$$m_1 - s = \epsilon_1 * f + \epsilon_2 * F \quad \text{and} \quad m_2 - t = \epsilon_1 * g + \epsilon_2 * G, \tag{8}$$

where $\epsilon_1 = A/q$ and $\epsilon_2 = a/q$ are polynomials whose coefficients are between $-1/2$ and $1/2$.

As m_1 and m_2 vary across all mod q polynomials, it is easy to check that A varies uniformly across all mod q polynomials, so to all intents and purposes, the coefficients of ϵ_1 may be treated as independent random variables that are uniformly distributed in the interval $(-1/2, 1/2)$. Hence on average we have $\|\epsilon_1\| \approx \sqrt{N/12}$. A similar remark applies to a and ϵ_2, so also $\|\epsilon_2\| \approx \sqrt{N/12}$.

We can now estimate the distance from (s, t) to (m_1, m_2) using $\|\epsilon_1\| \approx \|\epsilon_2\| \approx \sqrt{N/12}$ and the quasimultiplicativity of the norm:

$$\|(m_1 - s, m_2 - t)\|^2 = \|(\epsilon_1 f + \epsilon_2 F, \epsilon_1 g + \epsilon_2 G)\|^2 \approx \frac{c^2 N^3}{72} \left(1 + \frac{12}{N}\right). \tag{9}$$

Key creation The pair (f, g) which determines h, and the NTRU lattice, is created at random. The signer needs, however, a full (short) basis of the lattice $\mathcal{L}_h$ to produce a valid signature, and needs to create another pair (F, G) - short, independent of (f, g), and in $\mathcal{L}_h$. The general strategy for completing the basis of is to project f and g down to $\mathbb{Z}$ via the resultant mapping, which respects multiplication, i.e. we consider f and g as elements of $\mathbb{Z}[X]$ and find $\rho_f, \rho_g, k_f, k_g \in \mathbb{Z}[X]$ and $R_f, R_g \in \mathbb{Z}$ such that

$$\begin{aligned} \rho_f f + k_f(X^N - 1) &= R_f = \text{resultant}(f, X^N - 1), \\ \rho_g g + k_g(X^N - 1) &= R_g = \text{resultant}(g, X^N - 1). \end{aligned}$$

Assuming that R_f and R_g are coprime over the integers, we can now use the extended Euclidean algorithm to find $\alpha, \beta \in \mathbb{Z}$ such that $\alpha R_f + \beta R_g = 1$, in which case we have

$$(\alpha \rho_f) f + (\beta \rho_g) g = 1 + k(X^N - 1).$$

Theorem 2. *Let $F = -q\beta\rho_g$ and $G = q\alpha\rho_f$. The vectors $\{(f,g),(F,G)\}$ form an R-basis for the NTRU R-module $M_{h,q}$ generated by $\{(1,h),(0,q)\}$.*

Proof. Let $\mathcal{L}$ be the R-module generated by $\{(f,g),(F,G)\}$; we will show $M_{h,q} \subseteq \mathcal{L}$ and $\mathcal{L} \subseteq M_{h,q}$. An arbitrary $\mathbf{v} \in M_{h,q}$ can be written $\mathbf{v} = a(1,h) + b(0,q) = (a, ah + bq)$ for some $a, b \in R = \mathbb{Z}[X]/(X^N - 1)$. Since $h = f^{-1}g \bmod q$ we know that $(hf - g)/q \in R$ and similarly $(hF - G)/q = -h\beta\rho_g - \alpha\rho_f \in R$, thus $c = a(G - hF)/q - bF, d = (hf - g)/q + bf \in R$. It is a simple matter to confirm that $v = c(f,g) + d(F,G)$ so $M_{h,q} \subseteq \mathcal{L}$. To show the converse let $a = cf + dF, b = c(g - hf)/q + d(G - hF)/q \in R$.

A useful way to view $M_{h,q}$ is as a matrix of generating rows, in which case the above theorem can be seen as a unimodular change of basis.

$$\begin{pmatrix} f & g \\ F & G \end{pmatrix} = \begin{pmatrix} f & (g - fh)/q \\ F & (G - Fh)/q \end{pmatrix} \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}$$

With this notation we can define the *discriminant* of the R-module to be the determinant of the matrix, which can be seen to be an invariant modulo multiplication by a unit of R.

Since the definition of the resultant of f with $X^N - 1$ is the product of f evaluated at all the complex roots of $X^N - 1$, we know that $f(1) = d_f$ divides R_f. If we let ζ denote a *primitive root* of $X^N - 1$, then the remaining product R_d/d_f is actually the norm of $f(\zeta)$ when considered as an element of the field $\mathbb{Q}[\zeta]$. When $N - 1$ is reasonably composite the subfield structure[1] of $\mathbb{Q}[\zeta]$ may be utilised to yield a more efficient way of finding $\rho'_f \in \mathbb{Z}[X]$ and $R'_f = R_f/d_f \in \mathbb{Z}$ such that

$$\rho'_f f = R'_f \bmod \Phi(X).$$

In this case it is relatively easy to check that $\rho_f = d_f\rho'_f + k\Phi(X)$ where $k = (R'_f - d_f l)/N \in \mathbb{Z}$, and l is the sum of the coefficients of ρ'_f.

A problem with the F and G generated by these techniques is that although they do complete the basis, they typically have very large coefficients. However, we can clearly remove any R-multiple of the vector (f,g) from (F,G) - in fact, it suffices to reduce the first coordinate F. Suppose we find a k such that $\|F - kf\|$ is small. Then since $fG - gF = q$, we have

$$\left\| \frac{G}{g} - \frac{F}{f} \right\| = \left\| \frac{q}{fg} \right\| \approx \frac{q}{\|f\| \cdot \|g\|}$$

[1] It is an extremely interesting and open question to know if there are any security implications to using an N such that $N - 1$ is highly composite.

(For the parameter set $(N, q, d) = (251, 128, 72)$ this quantity is equal to 2.49, which means that on average, the coefficients of Gg^{-1} and Ff^{-1} differ by only 0.157.)

To minimize $\|F - kf\|$ we would like to choose k to be equal to $l = F/f \in \mathbb{Q}[X]/(X^N - 1)$, where we know $1/f = (1/R_f)\rho_f \in \mathbb{Q}[X]/(X^N - 1)$. However we are of course limited to take $k \in R$, so we take $k = \lfloor l \rceil$. The task of finding a k which minimizes the Euclidean norm of the coefficients of $F - kf$ can be viewed as a N-dimensional APPR-CVP lattice problem, and the procedure above is equivalent to Babai's "inverting and rounding" approach [Ba 86] The lattice for this N dimensional CVP problem is given by the circulant matrix generated by f. This is a well reduced lattice, which is essential for Babai's technique, since f is a relatively sparse binary element of R and thus highly orthogonal to its rotations.

In practice however one obtains a smaller result by treating the two components together. This corresponds to the standard lattice paradigm of multiplying a non-square basis by its transpose in order to be able to perform Babai's inverting and rounding technique. In this context the optimal k to reduce $\|(F, G) - k(f, g)\|$ turns out to be

$$k = \left\lfloor \frac{\bar{f}F + \bar{g}G}{f\bar{f} + g\bar{g}} \right\rceil,$$

where $\bar{f}(x) = f(1/x) \bmod X^N - 1$ and similarly for $\bar{g}$.

2.2 Hash function attacks

A signature is a close vector (s, t) to a an arbitrary point (m_1, m_2). From this signature it is easy to construct a signature for the point $(0, m_2 - m_1 - qv)$, namely $(s, t) - m_1(1, h) - v(0, q)$. So we shall only solve the CVP for points of the form $(0, m)$.

We now turn to security issues arising from the mapping of a digital document D to a message representative $(0, m)$.

This mapping is actually a two stage process. First a standard secure hash function H_1 is applied to D to give an output $H_1(D)$ consisting of β bits for an appropriate choice of β. Next a (public) function

$$H_2 : (\mathbb{Z}/2\mathbb{Z})^\beta \longrightarrow (\mathbb{Z}/q\mathbb{Z})^{2N}$$

is applied to $H_1(D)$ to yield the message digest $m = H_2(H_1(D))$. The function H_2 should map the 2^β possible H_1 hash values in a reasonably uniform manner into the set of q^{2N} possible message digests.

Two potential attacks arise related to this mapping. First, if two digital documents D and D' map to two message representatives m and m' which are very close together, and if the signer can be induced to sign both of them, then there is a small, but nontrivial, possibility that the difference of the signatures is a small element of the underlying lattice. This might reveal the private key [14]. Such a pair of documents would endanger all NTRUSIGN implementations using a common mapping H.

Another class of forgery attacks is also possible, arising from the fact that an attacker can generate arbitrarily many lattice points themselves of the form $(u, uh \bmod q)$ for aribtrary u. In this case, the attacker generates a large set $\mathcal{L}$ of lattice points and a large set $\mathcal{M}$ of message representatives, and checks to see if one of the lattice points in $\mathcal{L}$ signs one of the message representatives in $\mathcal{M}$.

To analyze both of these problems, we consider the collision-resistance of a uniform random mapping into $(\mathbb{Z}/q\mathbb{Z})^{2N}$. The measure of this resistance is the number of points that need to be generated by this mapping before there is a chance of greater than 50% that two image points exist within a distance B_{coll} of each other. To calculate this, consider where a mapping H_2 can place a succession of points. The first point can go anywhere. So long as the second point is at least B_{coll} from the first point, there is no collision. The third point needs to avoid (at worst) two balls of radius B_{coll}. Continuing with this reasoning, we find:

$$\begin{aligned}
&\text{Prob(no collision on } (n+1)\text{st random point)} \\
&\qquad \leq \prod_{k=0}^{n} \left(1 - k \cdot \frac{\text{Volume of an } N\text{-ball of radius } B_{\text{coll}}}{\text{Volume of an } N\text{-box of side } q}\right) \\
&\qquad = \prod_{k=0}^{n} (1 - k \cdot C), \qquad \text{where } C = \frac{\pi^{N/2}}{\Gamma(1+N/2)} \left(\frac{B_{\text{coll}}}{q}\right)^N.
\end{aligned}$$

In order for the probability of collision to be about $\frac{1}{2}$, therefore, we have the familiar birthday paradox formula $n_{\text{coll}} \geq \sqrt{1/C}$. Applying this to the key recovery attack, we find that for $(N, q) = (251, 128)$, an attacker will have to generate 2^{205} distinct messages before there is a 50% chance of finding two message representatives within $B_{\text{coll}} = 10$ of each other.

A similar analysis applies to the collision attack. Since each lattice point generated signs all points within a radius $\mathcal{N}$, an attacker who generates n lattice points can sign at most a fraction $n.C$ of all potential messages, where

$$C = \frac{\pi^{N/2}}{\Gamma(1+N/2)} \left(\frac{\mathcal{N}}{q}\right)^N.$$

If the attacker generates n points and k messages, her chance of not getting a collision is

$$(1 - nC)^k \approx 1 - knC.$$

To minimize $\max(k, n)$, we take $k = n$, and find that for the standard parameters $(N, q, \mathcal{N}) = (251, 128, 300)$, an attacker will have to generate 2^{86} lattice points and the same number of message representatives to have a 50% chance of forging a signature by this method.

3 Lecture 3

In this lecture, we discuss the security of NTRUSign from a number of different attacks. NTRUSign is a finite transcript scheme, but before analyzing transcript leakage, we'll deal with the more straightforward potential security problems.

3.1 Security of NTRUSign against forgery

Working only with the public parameters, and without the knowledge of any signed messages, key recovery is equivalent to finding small vectors in the NTRU lattice. Let's consider the difficulty of forging a specific signature.

Suppose a forger picks a small s, with the hope that $m - sh \bmod q$ will have all small coefficients too. On average these coefficients will be more-or-less random modulo q, so the average norm of an attempted forgery will be $q\sqrt{N/12}$. (This last quantity comes from an expected value computation: take a sequence of N independent random variables uniformly distributed in a q-interval, then the expected value of the sum of their squares is $q^2N/12$.) Since the asymptotic choices of d_f, d_g and q are all $O(N)$ we see that the forgery will have norm $O(N^{3/2})$, which is the same order of magnitude as the typical valid signature norm. Thus the security of NTRUSign lies in the relative constants involved.

A forgery attack may combine the preselection of some of the coordinates with lattice reduction techniques on a lower dimensional lattice to locate the remaining coordinates. This is the approach for convolution modular lattices developed by Gentry, Jonsson and Stern [7]. In this scenario, a forger preselects somewhat fewer than N coordinates and use lattice reduction techniques on a lower dimensional lattice to find the remaining coordinates. Let's say that αN of the coordinates of s and t are selected, for some choice of $0 \leq \alpha \leq 1$. Now lattice reduction techniques

on a lattice of dimension $(2-\alpha)N$ and determinant q^N to make the remaining $(1-\alpha)N$ coordinates as small as possible. Notice that $\alpha = 0$ corresponds to pure lattice reduction and $\alpha = 1$ corresponds to pure exhaustive search.

The actual lattice in [7] has determinant $q^{N(1+\alpha)}$ so we'll use this constant in the following computations. First observe that if $\alpha = 1$, the likelihood of an finding a valid signature (by solving $t = h * s$ for the remaining values) is just the probability that the sum of the squares of N integers chosen at random in the interval $[-q/2, q/2]$ will be less than the preassigned `Normbound`. An upper estimate for this probility is simply the volume of the appropriate ball divided by the volume of the q-cube. This probability can be made negligible for practical choices of the remaining parameters.

As α increases, the fundamental ratio (cf. (**??**))

$$\frac{\texttt{NormBound}}{\sqrt{\frac{(2-\alpha)N}{2\pi e}} \cdot q^{(1+\alpha)/(2-\alpha)}}$$

decreases, and when it passes below 1, the Gaussian heuristic says that it is very unlikely for any solutions to exist. For example, $\texttt{NormBound} = 310$ gives a value of $\alpha = 0.3835$, which corresponds to a lattice of dimension 405. Thus a lattice reduction attack cannot hope to be reduced below dimension 405 (down from 502). Further, as the dimension is reduced towards 405, the advantage gained from the reduction in dimension is at least partially eliminated due to the decrease in the Gauss ratio.

`NormBound`	α	Lattice Dim
300	0.3772	407
310	0.3835	405
320	0.3895	404
350	0.4062	400
380	0.4213	396
400	0.4305	393
420	0.4392	391
480	0.4624	385

Table 1. Minimum Usable Lattice Dimension — $N = 251$, $q = 128$

3.2 Transcript analysis attacks

Recall that the difference between the message to be signed (m_1, m_2) and signature satisfies

$$(m_1, m_2) - (s, t) = \begin{pmatrix} A/q & a/q \end{pmatrix} \begin{pmatrix} f & g \\ F & G \end{pmatrix}.$$

Since the coefficients of a and A are between $-q/2$ and $q/2$, it follows that

$$m_1 - s = \epsilon_1 * f + \epsilon_2 * F \quad \text{and} \quad m_2 - t = \epsilon_1 * g + \epsilon_2 * G, \tag{10}$$

where $\epsilon_1 = A/q$ and $\epsilon_2 = a/q$ are polynomials whose coefficients are between $-1/2$ and $1/2$. With $m_1 = 0$, and writing $m_2 = m$, we see that The polynomials A and a are related by

$$f * m = a + q * b \pmod{q} \qquad \text{and} \qquad -F * m = A + q * B. \tag{11}$$

Altogether, we see that $s = f * \epsilon_F - F * \epsilon_f$, where

$$\epsilon_F = \epsilon_F(m) = \left\{ \frac{F * m}{q} \right\}$$

and

$$\epsilon_f = \epsilon_f(m) = \left\{ \frac{f * m}{q} \right\},$$

and similarly for $t - m$.

Each of ϵ_F and ϵ_f are uniformly distributed in $[-1/2, 1/2]$, and so a transcript of signatures $s_1, ..., s_n$ reveals a list whose averages $(1/n) \sum s_i$ converge to 0. However, the averages of higher moments of the s_i's will converge to a nonzero quantity.

We first introduce some terminology. If $a(X) = \sum a_i X^i \in R$ be a polynomial, then the *reversal of* a is the polynomial

$$a^{\text{rev}}(X) = a(X^{-1}) = a_0 + \sum_{i=1}^{N-1} a_{N-i} X^i.$$

We then set

$$\bar{a}(X) = a(X) * a^{\text{rev}}(X) \in R_q = \frac{\mathbb{Z}_q[X]}{(X^N - 1)}.$$

Notice that $\bar{a}$ has the form

$$\bar{a} = \sum_{k=0}^{N-1} \left(\sum_{i=0}^{N-1} a_i a_{i+k} \right) X^k.$$

In particular, the constant term of $\bar{a}$ is $\sum a_i^2$, and if the coefficients of a are centered to have mean 0, then the constant term of $\bar{a}$ will tend to be considerably larger than the other terms. In the language of the Fourier transform, the DFT of the polynomial $a^{\text{rev}}(X)$ is simply the conjugate of the DFT of $a(X)$.

Consider the convergence of the second moment $\bar{s}_i = s_i * s_i^{\text{rev}}$. To find this value, we let s_m denote the signature on the document hash m. If X is any quantity that depends on m modulo q, we write $E(X)$ to denote the average value (or expecatation) as m ranges over R_q,

$$E(X) = \frac{1}{q^N} \sum_{m \in R_q} X_m.$$

Then the expectation of $\bar{s}_m$, which we denote by $\bar{s}_{\text{target}}$, is given by the formula

$$\begin{aligned}
\bar{s}_{\text{target}} &= E(\bar{s}_m) \\
&= E\left(\overline{f * \epsilon_F(m) + F * \epsilon_f(m)} \right) \\
&= E\left(\big(f * \epsilon_F(m) + F * \epsilon_f(m)\big) * \big(f * \epsilon_F(m) + F * \epsilon_f(m)\big)^{\text{rev}} \right) \\
&= \bar{f} * E\left(\overline{\epsilon_F(m)} \right) + \bar{F} * E\left(\overline{\epsilon_f(m)} \right) \\
&\qquad + f * F^{\text{rev}} * E\left(\epsilon_F(m) * \epsilon_{f^{\text{rev}}}(m^{\text{rev}}) \right) + F * f^{\text{rev}} * E\left(\epsilon_f(m) * \epsilon_{F^{\text{rev}}}(m^{\text{rev}}) \right)
\end{aligned} \tag{12}$$

The first two terms are the most straightforward to evaluate.

Proposition 1. *Assume that f is invertible in R/qR and that q is even. Then*

$$E\left(\overline{\epsilon_f(m)} \right) = \frac{N}{12} \left(1 - \frac{1}{q^2} + \frac{3}{q^2} \sum_{k=0}^{N-1} X^k \right). \tag{13}$$

Proof. Note that $\epsilon_f(m) = \{f * m/q\}$ depends only on m modulo q. So if we make the change of variables $m = f^{-1} * m'$, where $f^{-1} \in R$ is an inverse of f in R_q and m' runs over R_q, the value of the sum (13) does not change. It thus suffices to consider the case $f = 1$, in which case $\epsilon_1(m)$ is simply m/q.

We are assuming that q is even, since this is the case in which we are principally interested. For notational convenience we set $Q = q/2$.

We may treat the coefficients of $m(X)$ as independent random variables that are uniformly distributed in the set $\{-Q, -Q+1, \dots, Q-1\}$. Then

$$E(m_i) = \frac{1}{q}\sum_{u=-Q}^{Q-1} u = -\frac{1}{2} \quad \text{and} \quad E(m_i^2) = \frac{1}{q}\sum_{u=-Q}^{Q-1} u^2 = \frac{q^2+2}{12},$$

and therefore,

$$\begin{aligned} E(\overline{\epsilon_1(m)}) &= \frac{1}{q^2}\sum_{k=0}^{N-1}\sum_{i=0}^{N-1} E(m_i m_{i+k})X^k \\ &= \frac{1}{q^2}\sum_{i=0}^{N-1} E(m_i^2) + \frac{1}{q^2}\sum_{k=1}^{N-1}\sum_{i=0}^{N-1} E(m_i)E(m_{i+k})X^k \\ &= \frac{N(q^2+2)}{12q^2} + \frac{N}{4q^2}\sum_{k=1}^{N-1} X^k. \end{aligned}$$

The evaluation of the cross terms in the formula for $\bar{s}_{\text{target}}$ would be similarly straightforward if the quantities ϵ_f and ϵ_F were independent, but they are not. However, it is still possible to give a simple formula for the expectation.

Proposition 2. *Let $q = p^r$ be a prime power, and let $f, g \in R$ by any two polynomials with the property that*

$$f^{-1} \bmod q \text{ exists} \qquad \text{and} \qquad f^{-1} * g \not\equiv cX^k \pmod{p}. \tag{14}$$

(In other words, $f^{-1} * g$ is not a monomial modulo p.) Then

$$E\big(\epsilon_f(m) * \epsilon_g(m)\big) = \begin{cases} 0 & \text{if } p \text{ is odd,} \\ \frac{1}{4}(1 + X + \cdots + X^{N-1}) & \text{if } p = 2. \end{cases} \tag{15}$$

Proof. We first observe that $\epsilon_f(m)$ only depends on f modulo q and that for any polynomial h that is invertible modulo q, we have

$$E\big(\epsilon_{f*h}(m) * \epsilon_{g*h}(m)\big) = E\big(\epsilon_f(m) * \epsilon_g(m)\big).$$

Taking $h = f^{-1} \bmod q$, it thus suffices to treat the case $f = 1$. Then $\epsilon_f(m) = m/q$, and our assumption is that g is not a monomial.

The degree k coefficient of the expection is equal to

$$\frac{1}{q^N} \sum_m \sum_{i=0}^{N-1} \frac{m_{k-i}}{q} \left\{ \left(\frac{g * m}{q} \right)_i \right\}.$$

For any particular i, we consider the sum

$$\sum_m m_{k-i} \left\{ \left(\frac{g * m}{q} \right)_i \right\} = \sum_m m_{k-i} \left\{ \frac{1}{q} \sum_{j=0}^{N-1} g_j m_{i-j} \right\}.$$

The fact that g is not a monomial means that we can find an index ℓ with $\ell \neq k - i$ and $g_{i-\ell} \not\equiv 0 \pmod{p}$. In the outer sum over $m = (m_0, m_1, \ldots, m_{N-1})$, we consider the sum over the variable m_ℓ. This sum can be moved past the outer m_{k-i}, so we obtain an inner sum that looks like

$$\sum_{m_\ell \bmod q} \left\{ \frac{1}{q} \sum_{j=0}^{N-1} g_j m_{i-j} \right\}. \tag{16}$$

Since the fractional part only depends on the value of the numerator modulo q, we can make a change of variables in the outer sum by setting $m_\ell = g_{i-\ell}^{-1} u \pmod{q}$. Then the sum becomes

$$\sum_{u \bmod q} \left\{ \frac{1}{q} \left(u + \sum_{j \neq i-\ell} g_j m_{i-j} \right) \right\}.$$

As u ranges over all values mod q, so does $u + \sum g_j m_{i-j}$, so the above sum is simply equal to

$$\sum_{w \bmod q} \left\{ \frac{w}{q} \right\} = -\frac{q}{2}.$$

(This is assuming that q is even. If q is odd, then the sum will be symmetric and will equal 0.)

These computations show that from the average of $\bar{s}$ over a moderately long transcript, the value of

$$\bar{F} + \bar{f}$$

can probably be recovered. So we assume that the attacker, in posession of a transcript of signatures, knows this value exactly.

Now consider the average of the quantities $\bar{s}_i^2$, and denote the (experimental) average value of $\bar{s}^2$ over the finite transcript by

$$S_{\text{exper}} = \frac{1}{n} \sum_{i=1}^{n} \bar{s}_i^2.$$

As m varies over all possible N-tuples in $\mathbb{Z}_q$, the average, as $n \to \infty$, will be denoted S_{target}.

Proposition 3. *The expectation of $\bar{s}^2$ has the form*

$$S_{target} = E(\bar{s}^2) = E_2 * \bar{f}^2 + 4 * E_1^2 * \bar{f} * \bar{F} + E_2 * \bar{F}^2, \tag{17}$$

where E_1 and E_2 are given by the formulas:

$$E_1 = \alpha_1 + \beta_1(X + X^2 + \cdots + X^{N-1})$$
$$E_2 = \alpha_2 + \beta_2(X + X^2 + \cdots + X^{N-1})$$
$$\alpha_1 = N\left(\frac{1}{12} + \frac{1}{6q^2}\right)$$
$$\beta_1 = \frac{N}{4q^2}$$
$$\alpha_2 = \frac{N^2 - N}{72}\left(1 - \frac{1}{q^2}\right)^2 + N\left(\frac{1}{80} - \frac{1}{24q^2} + \frac{7}{240q^4}\right) + \frac{N^2}{8q^2} - \frac{N^2}{8q^4} + \frac{N^3}{16q^4}$$
$$\beta_2 = \frac{N}{144}\left(1 - \frac{1}{q^2}\right)^2 + \frac{N^2}{8q^2}\left(1 - \frac{1}{q^2} + \frac{N}{2q^2}\right)$$

Equivalently, S_{target} is given by the formula

$$2S_{target} = (E_2 + 2E_1^2)(\bar{F} + \bar{f})^2 + (E_2 - 2E_1^2)(\bar{F} - \bar{f})^2. \tag{18}$$

Proof. The proof of the formula (17) for $E(\bar{s}^2)$ is a calculation similar to the calculation of $E(\bar{s})$ done in Section **??**, but the computation is considerably more complicated. We omit the details. The equivalence of (17) and (18) is obvious.

This calculation is similar to, but more complicated than, the calculation for $E(\bar{s})$.

A rough approximation of the coefficients of $2S_{\text{target}}$ using $E_1 \approx N/12$ and $E_2 \approx N^2/72$ gives

$$\|E_2 + 2E_1^2\| \approx N^2/36 \qquad \text{and} \qquad \|E_2 - 2E_1^2\| \approx N/120.$$

For $N = 251$ and $q = 128$ we find the exact values

$$\|E_2 + 2E_1^2\| \approx 1744.24 \qquad \text{and} \qquad \|E_2 - 2E_1^2\| \approx 2.09.$$

The attackers objective is to find information about the private key - specifically, the first goal is to approximate $\bar{f}$ using the experimental average S_{exper}.

Consider the expression

$$A := \frac{2S_{\text{exper}} - (E_2 + 2E_1^2)(\bar{F} + \bar{f})^2}{E_2 - 2E_1^2} = (\bar{F} - \bar{f})^2 - \frac{S_{\text{target}} - S_{\text{exper}}}{E_2 - 2E_1^2}. \quad (19)$$

If the second term were small in comparison to the first term, then in principle, one could take the square root to obtain

$$B := A^{1/2} \approx (\bar{F} - \bar{f}) - \frac{S_{\text{target}} - S_{\text{exper}}}{2(\bar{F} - \bar{f})(E_2 - 2E_1^2)}. \quad (20)$$

Subtracting this from the known value of $\bar{F} + \bar{f}$ gives

$$C := \frac{(\bar{F} + \bar{f}) - B}{2} \approx \bar{f} + \frac{S_{\text{target}} - S_{\text{exper}}}{4(\bar{F} - \bar{f})(E_2 - 2E_1^2)}. \quad (21)$$

The quantity C is an ϵ-approximation to f, on average, if

$$\|C - \bar{f}\| \leq \epsilon\sqrt{N}. \quad (22)$$

We will assume that S_{exper} has been adjusted to have the same mean and centered norm as S_{target}, i.e., they have the same mean and variance. Then we can use the following formula to approximate the error:

$$\|v - w\|^2 = \big(\|v\| - \|w\|\big)^2 + 2\|v\| \cdot \|w\|\big(1 - \rho(v, w)\big).$$

Here $\rho(v, w)$ is the correlation coefficient of v and w. Using this formula and the normalization $\|S_{\text{exper}}\| = \|S_{\text{target}}\|$, we find that

$$\|C - \bar{f}\| \approx \left\| \frac{S_{\text{target}}}{(\bar{F} - \bar{f})(E_2 - 2E_1^2)} \right\| \sqrt{\frac{1 - \rho(S_{\text{target}}, S_{\text{exper}})}{2}}.$$

Hence C will be an ϵ approximation of f if if S_{exper} satisfies

$$\left\| \frac{S_{\text{target}}}{(\bar{F} - \bar{f})(E_2 - 2E_1^2)} \right\| \sqrt{\frac{1 - \rho(S_{\text{target}}, S_{\text{exper}})}{2}} \leq \epsilon\sqrt{N}.$$

Equivalently, the correlation between S_{target} and S_{exper} must satisfy

$$\rho(S_{\text{target}}, S_{\text{exper}}) \geq 1 - 2\epsilon^2 N \left\| \frac{S_{\text{target}}}{(\bar{F} - \bar{f})(E_2 - 2E_1^2)} \right\|^{-2}.$$

For the particular parameters $N = 251$ and $q = 128$ and a sample key (f, F), an easy calculation gives

$$\left\| \frac{S_{\text{target}}}{(\bar{F} - \bar{f})(E_2 - 2E_1^2)} \right\| \approx 10^{5.847}.$$

Hence our theory predicts that the correlation $\rho = \rho(S_{\text{target}}, S_{\text{exper}})$ and the average coefficient error $\epsilon = \|C - \bar{f}\|/\sqrt{N}$ should be related by the formula

$$\rho \approx 1 - 10^{-9}\epsilon^2,$$

or equivalently

$$\epsilon \approx 10^{4.5}\sqrt{1 - \rho}. \tag{23}$$

To reconstruct $\bar{f}$ from C, an attacker will need $\epsilon < 0.5$, and possibly even smaller. We conducted an experiment to determine the accuracy of the theoretical prediction and to estimate the average coefficient error. We set S_{exper} equal to S_{target} plus a small error and tabulated how many of the coefficients of the computed value C differed from the corresponding coefficient of its target value $\bar{f}$. The experiments showed that direct recovery of f was unlikely to be successful if $1 - \rho$ is greater than 10^{-10}, but would almost certainly succeed if $1 - \rho$ were smaller than 10^{-13}.

We implemented the above algorithm and used it on long transcripts to attempt to recover $\bar{f}$. After 10 million signatures, the correlation between C and $\bar{f}$ was only 43.1%, and the average coefficient of $C - \bar{f}$ had magnitude 4.99. Since the average coefficient of $\bar{f}$ itself had magnitude 4.68, this would not appear to be helpful.

Extending the transcript to 100 million signatures improved matters a bit. The correlation between C and $\bar{f}$ was 74.2%, but the average coefficient of $C - \bar{f}$ was still 3.36.

Remark 8. Because a finite transcript of some (large) length leaks private information, it is important to consider measures which lengthen the size of a practical transcript attack. One such measure involves perturbing the lattice point m. Specifically, after hashing the digital document D to obtain the lattice point m, one could add a varying and secret ϵ to m obtaining m'. If ϵ is sufficiently small, the signature for m is also a valid signature for m'. The distribution for ϵ should be chosen with care,

since there are subtle attacks which are based on discerning the difference between the distribution of normal signatures and the distributions of the perturbed signatures. (In fact, it was attacks of this sort that Gentry and Szydlo exploited in demonstrating the weakness of the scheme NSS.)

References

1. M. Ajtai, C. Dwork, *A public-key cryptosystem with worst case/average case equivalence*. In Proc. 29th ACM Symposium on Theory of Computing, 1997, 284–293.
2. D. Coppersmith and A. Shamir, *Lattice Attacks on NTRU*, Proceedings Eurocrypt '97, Lecture Notes in Computer Science, Springer-Verlag, 1997
3. L. Babai *On Lovász lattice reduction and the nearest lattice point problem*, Combinatorica, vol. 6, 1986, 1–13
4. Craig Gentry, Jakob Jonsson, Jacques Stern, Michael Szydlo *Cryptanalysis of the NTRU Signature Scheme (NSS) from Eurocrypt 2001*, Advances in Cryptology—Asiacrypt '01, Lecture Notes in Computer Science, Springer-Verlag, 2001.
5. A.K. Lenstra, H.W. Lenstra Jr., L. Lovász, *Factoring polynomials with rational coefficients*, Mathematische Ann. 261 (1982), 513–534.
6. D. Micciancio, *The shortest vector in a lattice is hard to approximate to within some constant*, Proc. 39th Symposium on Foundations of Computer Science, 1998, 92–98.
7. Craig Gentry, Jakob Jonsson, Jacques Stern, Michael Szydlo *Cryptanalysis of the NTRU Signature Scheme (NSS) from Eurocrypt 2001*, Advances in Cryptology—Asiacrypt '01, Lecture Notes in Computer Science, Springer-Verlag, 2001. bibitemGentrySz C. Gentry,M Szydlo*Cryptanalysis of the Revised NTRU Signature Scheme* Advances in Cryptology—Eurocrypt '02, Lecture Notes in Computer Science, Springer-Verlag, 2002.
8. O. Goldreich, S. Goldwasser, S. Halevi, *Public-key cryptography from lattice reduction problems*. In Proc. CRYPTO'97, Lect. Notes in Computer Science 1294, Springer-Verlag, 1997, 112–131.
9. J. Hoffstein, J. Pipher, J.H. Silverman, *NTRU: A new high speed public key cryptosystem*, in Algorithmic Number Theory (ANTS III), Portland, OR, June 1998, Lecture Notes in Computer Science 1423 (J.P. Buhler, ed.), Springer-Verlag, Berlin, 1998, 267–288.
10. J. Hoffstein, N. Howgrave-Graham, J. Pipher, J.H. Silverman, W. Whyte *NTRUSign: Digital signatures using the NTRU lattice. Preliminary draft 2* `http://www.ntru.com/NTRUFTPDocsFolder/NTRUSign_v2.pdf`
11. J. Hoffstein, J. Pipher, J.H. Silverman, *NSS: An NTRU Lattice-Based Signature Scheme*, Advances in Cryptology—Eurocrypt '01, Lecture Notes in Computer Science, Springer-Verlag, 2001.
12. J. Hoffstein, D. Lieman, J.H. Silverman, *Polynomial Rings and Efficient Public Key Authentication*, in Proceeding of the International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC '99), Hong Kong, (M. Blum and C.H. Lee, eds.), City University of Hong Kong Press.

13. J. Hoffstein, J.H. Silverman, *Polynomial Rings and Efficient Public Key Authentication II*, in Proceedings of a Conference on Cryptography and Number Theory (CCNT '99), (I. Shparlinski, ed.), Birkhauser.
14. T. Meskanen and A. Renvall, University of Turku, private communication.
15. Phong Nguyen and David Pointcheval, emphAnalysis and Improvements of NTRU Encryption Paddings, preprint, www.di.ens.fr/ pnguyen,pointche
16. P. Nguyen, *Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto '97*, Advances in Cryptology—Proceedings of CRYPTO '99, (August 15–19, 1999, Santa Barbara, California), M. Wiener (ed.), Lecture Notes in Computer Science, Springer-Verlag.
17. P. Nguyen and J. Stern, *Lattice Reduction in Cryptology: An Update*, ANTS 2000, pp 85-112.
18. A. Shamir, *A polynomial-time algorithm for breaking the basic Merkel-Hellman cryptosystem*. In Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science, IEEE, 1982, 145–152.