Key Recovery and Message Attacks on NTRU-Composite

Craig Gentry

DoCoMo Communications Laboratories USA, Inc.
181 Metro Dr., San Jose, CA 95110, USA
cgentry@dcl.docomo-usa.com

Abstract. NTRU is a fast public key cryptosystem presented in 1996 by Hoffstein, Pipher and Silverman of Brown University. It operates in the ring of polynomials $\mathbb{Z}[X]/(X^N - 1)$, where the domain parameter N largely determines the security of the system. Although N is typically chosen to be prime, Silverman proposes taking N to be a power of two to enable the use of Fast Fourier Transforms. We break this scheme for the specified parameters by reducing lattices of manageably small dimension to recover partial information about the private key. We then use this partial information to recover partial information about the message or to recover the private key in its entirety.

1 Introduction

NTRU is a fast public key cryptosystem that operates in the ring of truncated polynomials given by $\mathbb{Z}[X]/(X^N-1)$, where the domain parameter N largely determines the security of the system. Typically N is chosen to be a prime number (not for security reasons, but because having N prime maximizes the probability that the private key has an inverse with respect to a specified modulus [11]). Recently, however, Silverman has proposed taking N to be a power of two to allow the use of Fast Fourier Transforms when computing the convolution product of elements in the ring [13].

In this paper, we present lattice-based attacks that are especially effective when N is composite. We show how to use low-dimensional lattices to find a folded version of the private key, where the folded private key has d coefficients with d dividing N. This folded private key can be used either to obtain a folding of the plaintext message, or as partial information to help us recover the entire private key. Using this attack, we were able to recover entire private keys for the NTRU-256 scheme proposed by Silverman in an average of about 3 minutes.

2 Notation

We denote the ring of integers by $\mathbb{Z}$, and the ring of integers modulo q by $\mathbb{Z}_q$, which are taken in the interval $(-\frac{q}{2}, \frac{q}{2}]$. The polynomial ring $\mathbb{Z}_q[X]/(X^n - 1)$ contains all polynomials with degree less than n and coefficients in $\mathbb{Z}_q$. The

B. Pfitzmann (Ed.): EUROCRYPT 2001, LNCS 2045, pp. 182–194, 2001.

inverse of a polynomial $f \in \mathbb{Z}_q[X]/(X^n - 1)$ is denoted by f_q^{-1}. A polynomial may be described as a row vector:

$$f = (f_0, f_1, \ldots, f_{n-1}) = \sum_{i=0}^{n-1} f_i X^i \ .$$

Concatenation of f and g is denoted by (f, g). The convolution $f * g$ of two vectors is analogous to ordinary polynomial multiplication over $\mathbb{Z}[X]/(X^n - 1)$:

$$(f * g)_k = \sum_{i+j=k \bmod n} f_i g_j \ .$$

When d divides n, the d-dimensional folded version of f is defined by:

$$f_{(d)} = (\sum_{i=0 \bmod d}^{0 \le i < n} f_i, \sum_{i=1 \bmod d}^{0 \le i < n} f_i, \ldots, \sum_{i=d-1 \bmod d}^{0 \le i < n} f_i) \ .$$

In algebraic terms, $f_{(d)}$ may be described as the image of f under the canonical mapping from $\mathbb{Z}[X]/(X^n - 1)$ to $\mathbb{Z}[X]/(X^d - 1)$. The ith term of $f_{(d)}$ will be denoted $f_{(d),i}$. The circulant matrix associated with f is given by F, where:

$$F_{ij} = f_{j-i \bmod n} \ .$$

F_i will denote the ith row vector of F. $F_{(d)}$ will denote the circulant associated with $f_{(d)}$, and $F_{(d),i}$ will denote its ith row vector. $I_{(d)}$ will refer to the d-dimensional identity matrix.

3 The NTRU Cryptosystem

Public Parameters. The basic objects of the NTRU Cryptosystem are polynomials from the ring $\mathbb{Z}[X]/(X^N - 1)$, where N is a public parameter. Also public are two moduli, p and q, with g.c.d.$(p, q) = 1$ and $p \ll q$. For example, $(N, p, q) = (167, 3, 128)$ has been proposed as a high security parameter set [7]. Additional public parameters include S_f, S_g, S_m, and S_ϕ, which describe the space of allowable polynomials for private keys f and g, the plaintext message m, and a random polynomial ϕ that the sender uses in encrypting the message. These spaces are designed to limit f, g, m, and ϕ to vectors that have short Euclidean length (in practice, less than $\sqrt{N}$) and that typically are also very short in the l_∞-norm — i.e., the magnitudes of the individual coefficients are typically very small in relation to q. For example, in NTRU-167, S_f might limit f to those polynomials having exactly 61 coefficients equal to 1, 60 coefficients equal to -1, and 46 coefficients equal to 0 [7]. S_m always restricts the coefficients of m to $\mathbb{Z}_p$.

Key Creation. Choose random $f \in S_f$ and $g \in S_g$. Compute f_q^{-1} and publish the polynomial

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

as the public key. Both f and g are private, with f serving as the private key.

Encryption. Choose random ϕ in S_ϕ and compute the ciphertext:

$$e = m + p\phi * h \pmod{q} .$$

Decryption. Compute

$$\begin{aligned} f * e &= f * m + p\phi * f * h \pmod{q} \\ &= f * m + p\phi * g \pmod{q} , \end{aligned}$$

where the second equality follows from the definition of h. Assuming S_f, S_m, S_ϕ, and S_g are chosen wisely, such that the coefficients of f, m, ϕ, and g are very small in relation to q, then, with high probability, we get

$$f * m + p\phi * g \pmod{q} = f * m + p\phi * g ,$$

which is to say that reduction modulo q has no effect. This is because the coefficients of $f * m + p\phi * g$, with high probability, already lie in $(-\frac{q}{2}, \frac{q}{2}]$ before reduction modulo q. Possessing the unreduced value of $f * m + p\phi * g$, we can compute

$$f * m + p\phi * g \pmod{p} = f * m \pmod{p} ,$$

and then

$$f_p^{-1} * f * m \pmod{p} = m \pmod{p} .$$

4 Previous Lattice Attacks on NTRU

Lattice attacks on NTRU (including our attack) have focused primarily on the following "key recovery" problem: find the private key f using only the public key h and public information about how f and g are chosen (S_f and S_g).[1] By the definition of h, we know that $f * h = g \pmod{q}$, but this information alone is clearly insufficient to recover f. Indeed, the set of pairs $u, v \in \mathbb{Z}^N$ that satisfy $u * h = v \pmod{q}$ is an additive abelian group of infinite cardinality. Even if we limit ourselves to pairs $u, v \in \mathbb{Z}_q^N$, we are still left with q^N distinct (u, v) pairs corresponding to the q^N distinct values that u can assume. How do we find the pair (f, g) from among these q^N possibilities?

We know that, to enable error-free decoding, the coefficient vectors of f and g each have short Euclidean length (less than $\sqrt{N}$ in current NTRU implementations). They are considerably shorter than the typical "random" N-dimensional vector with coefficients in $\mathbb{Z}_q$, which has an expected length of more than $\frac{q}{4}\sqrt{N}$.

[1] Non-lattice-based cryptanalysis of NTRU includes a meet-in-the-middle attack found by Odlyzko [12] and a chosen-ciphertext attack presented by Jaulmes and Joux [5], which exploited NTRU's inappropriate use of OAEP-like padding. We understand that NTRU now uses the hybridization method presented by Fujisaki and Okamoto [2] to obviate chosen-ciphertext attacks.

Also, one can show that it is extremely unlikely that the abelian group generated by a "randomly" chosen h' has a (u, v) pair as short as (f, g).[2] These facts may lead us to hypothesize that (f, g) is, in fact, the shortest nonzero vector $(u, v) \in \mathbb{Z}^{2N}$ such that $u * h = v \pmod{q}$. If this hypothesis is true, and if we can find an efficient way to find the shortest vector belonging to the group of (u, v) pairs, we can recover the private key. This provides the motivation for representing the group of (u, v) pairs as a "lattice," and then using "lattice basis reduction."

A "lattice" is a discrete additive subgroup of $\mathbb{R}^n$. For example, $\mathbb{Z}^n$ is a lattice. Also, the set of (u, v) pairs is a lattice, being an additive subgroup of $\mathbb{Z}^{2N}$. An equivalent, but more concrete, definition is that a lattice L consists of all integer linear combinations of some set of m linearly independent vectors $B = \{b_0, b_1, \ldots, b_{m-1}\}$, $b_i \in \mathbb{R}^n$. Here, m is the "dimension" of L, and B is called a "basis" of L. The basis B can be compactly represented by an $m \times n$ matrix where the ith row is the "basis vector" b_i, in which case L consists of the vectors that can be expressed as integer linear combinations of the rows of B. Bases for a lattice are not unique, but are related by unimodular transformations — i.e., if U is an integral $m \times m$ matrix with determinant ± 1, then UB is an equally valid basis for L. Typically, the goal of "lattice basis reduction" is to find a basis B' for L in which the basis vectors are as short as possible (usually in the Euclidean sense), with the basis vector b'_0 being the shortest nonzero vector in the entire lattice (allowing for possible ties).

Coppersmith and Shamir [1] give us the following explicit basis for the lattice of (u, v) pairs (recall that H denotes the circulant matrix corresponding to the public key h):

$$L_{CS} = \begin{bmatrix} I_{(N)} & H \\ 0 & qI_{(N)} \end{bmatrix} .$$

To see that any pair $(u, v) \in \mathbb{Z}^{2N}$ for which $u * h = v \pmod{q}$ is contained in the lattice generated by L_{CS}, let $a \in \mathbb{Z}^N$ be such that $u * h = v + qa$. Then, if we left-multiply L_{CS} by $(u, -a)$, we obtain (u, v). As a consequence, the private key pair (f, g) is an integer linear combination of the rows of L_{CS}. If (f, g) is actually the shortest vector in the generated lattice, as we have reason to believe,[3] then an "SVP-oracle" — a magical device which gives us the answer to the "shortest vector problem" *in a reasonable (polynomial) amount of time* — would give us the private key when given L_{CS} as input.

For the attacker, the problem is that actual lattice basis reduction algorithms, such as LLL and its variants, do not behave like SVP-oracles. The original LLL algorithm terminates in time polynomial in the dimension n of the lattice, but it is only guaranteed to find a vector that is no more than $2^{(n-1)/2}$ times — $2^{(2N-1)/2}$ times, in the case of L_{CS} — as long as the shortest vector. Obviously, such an algorithm is useless to us, considering that it is trivial to find vectors only about $\frac{q}{4}$ times as long as (f, g), as suggested above, and even these are far too long to be useful for decryption. Variants of LLL exist that find shorter

[2] See Appendix A.1.
[3] See Appendix A.1.

vectors, but they naturally have greater time-complexity. In particular, Schnorr defines a family of LLL-variants whose performances depend on a parameter called the "blocksize." Little is known about the average-case complexity of these variants, but it appears, based on numerous experiments by the authors of NTRU using Shoup's NTL library [8], that the time necessary to find (f, g) in the lattice grows at least exponentially in N (because the block size required for LLL to find (f, g) grows roughly linearly in N [3], and the running time of LLL is exponential in the block size [10]). The authors of NTRU estimate that, for $N > 90$, it takes current lattice reduction algorithms $e^{.2002N-7.608}$ seconds to find (f, g) on a 400 MHz machine, which translates into 4.607×10^{14} MIPS-years to break NTRU-263 [10].[4]

5 Cryptanalysis of NTRU-Composite

The problem with previous lattice-based attacks is that the dimension of the lattices involved is too high, given that the running time of LLL to return the target vector of these lattices is empirically exponential in the lattice dimension. Ideally, we would like to construct much smaller (and more easily reduceable) lattices whose shortest vectors contain at least some useful cryptanalytic information. We can do this if N is composite.

Theorem 1. *Let N be composite, and d be a nontrivial divisor. The mapping $\theta : \mathbb{Z}[X]/(X^N - 1) \to \mathbb{Z}[X]/(X^d - 1)$ given by*

$$\theta(f) = f_{(d)}$$

is a ring homomorphism.

Although this is a basic algebraic result, arising from the fact that $(X^d - 1)$ divides $(X^N - 1)$ when d divides N, we prove multiplication in a concrete fashion.

Proof.

$$\begin{aligned} g_{(d),k} &= \sum_{i=k \bmod d}^{0 \le i < N} g_i \\ &= \sum_{i=k \bmod d}^{0 \le i < N} \left(\sum_{x+y=i \bmod N}^{0 \le x,y < N} f_x h_y \right) \\ &= \sum_{x+y=k \bmod d}^{0 \le x,y < N} f_x h_y \\ &= \sum_{v+w=k \bmod d}^{0 \le v,w < d} \left(\left(\sum_{x=v \bmod d}^{0 \le x < N} f_x \right) \left(\sum_{y=w \bmod d}^{0 \le y < N} h_y \right) \right) \end{aligned}$$

[4] Refinements to L_{CS} by May [6] have made it possible to recover an NTRU-107 private key in 12 to 24 hours on a single 400 MHz machine [10], but do not seriously affect the security estimates for higher security levels, such as NTRU-167 [9].

$$= \sum_{v+w=k \bmod d}^{0 \le v,w < d} f_{(d),v} h_{(d),w} \ .$$

This gives us $f_{(d)} * h_{(d)} = g_{(d)}$, which is what we wanted. □

5.1 A Smaller Version of L_{CS}

With the equation $f_{(d)} * h_{(d)} = g_{(d)}$ in mind, we construct the following $2d$-dimensional analog of L_{CS} (recall that $H_{(d)}$ is the circulant corresponding to $h_{(d)}$):

$$L_{(d)} = \begin{bmatrix} I_{(d)} & H_{(d)} \\ 0 & qI_{(d)} \end{bmatrix} \ .$$

This lattice contains the vector $(f_{(d)}, g_{(d)})$. Notice that if N/d is not too large, then the smallness of the coefficients of f and g ensures that the coefficients of $f_{(d)}$ and $g_{(d)}$, each of which is a summation of N/d coefficients of f and g respectively, are also small. Assuming $(f_{(d)}, g_{(d)})$ is the shortest vector in $L_{(d)}$, we can find it using lattice reduction. We can then recover significant partial information about the private key by reducing a lattice whose dimension is only a fraction of the dimension of the lattice generated by L_{CS}.

In Appendix A.2, we give a tight upper bound on the length of $(f_{(d)}, g_{(d)})$ and show that, assuming f and g are "random" in a specified way, the expected length of $(f_{(d)}, g_{(d)})$ is equal to the length of (f, g) (once certain modifications are made to these vectors). This leads us to conclude that, at least when $d > \sqrt{N}$, $(f_{(d)}, g_{(d)})$ is almost certainly the shortest vector in $L_{(d)}$ for the same reasons that (f, g) is almost certainly the shortest vector in L_{CS}.

Remark: In the discussion above, we have limited our focus to homomorphisms of the form $\theta : \mathbb{Z}[X]/(X^N - 1) \to \mathbb{Z}[X]/(X^d - 1)$ and the folded lattices derived therefrom, but this need not be the case. More generally, we could consider homomorphisms of the form $\alpha : \mathbb{Z}_q[X]/(X^N - 1) \to \mathbb{Z}_q[X]/s(X)$ given by $\alpha(f) = f + < s(X), q >$, where $s(X)t(X) = (X^N - 1) \pmod q$ for some $t(X)$. However, such homomorphisms appear to be useful only when $(\alpha(f), \alpha(g))$ is a short vector that can be found using lattice basis reduction, and $(\alpha(f), \alpha(g))$ is always short only if $s(X)$ is an *extremely* short vector, preferably with a minimum of high degree coefficients (e.g., $(X^d - 1)$). Useful alternative homomorphisms therefore appear to be rare.

5.2 Message Attacks

Once we find $f_{(d)}$, we can make immediate use of it to recover the folded plaintext. Since folding is a ring homomorphism, we get:

$$f_{(d)} * e_{(d)} = f_{(d)} * m_{(d)} + p\phi_{(d)} * f_{(d)} * h_{(d)} = f_{(d)} * m_{(d)} + p\phi_{(d)} * g_{(d)} \pmod q.$$

We then proceed through the steps of decryption in the usual way until we obtain $m_{(d)}$. If $N/d = 2$, for example, knowing $m_{(d)}$ is tantamount to knowing

$m_i + m_{i+d}$ for $0 \leq i < d$, where the m_i are coefficients from the original plaintext. This could be useful information.

However, folding entails an increased likelihood of decryption errors, since the expected magnitudes of the coefficients of $f_{(d)} * m_{(d)} + p\phi_{(d)} * g_{(d)}$ are larger than those of $f * m + p\phi * g$ by a factor of $\sqrt{N/d}$. So, this message attack appears to be practical only for very small values of N/d.

5.3 Key Recovery Attacks

Alternatively, we can use $f_{(d)}$ to help us recover f. The basic concept behind this attack is the Chinese Remainder Theorem, which tells us, for example, that f is completely determined by the values of $f \pmod{X^d - 1}$ and $f \pmod{(X^N - 1)/(X^d - 1)}$.[5] Instead of using the lattice corresponding to $f \pmod{(X^N - 1)/(X^d - 1)}$, however, we use a different lattice with a shorter target vector.

Supposing, for example, that $N/d = 2$, we obtain linear equations of the form $f_{i+d} = f_{(d),i} - f_i$, so that we have

$$f = (f_0, f_1, \ldots, f_{d-1}, f_{(d),0} - f_0, f_{(d),1} - f_1, \ldots, f_{(d),d-1} - f_{d-1}).$$

Recall that in the lattice generated by L_{CS}, the target vector

$$(f, g) = \sum_{i=0}^{N-1} f_i(I_i, H_i) \pmod{q} ,$$

where (I_i, H_i) denotes the concatenation of the ith rows of the identity matrix and the circulant H. Using the dependencies in f, we obtain

$$\begin{aligned}(f, g) &= \sum_{i=0}^{d-1} f_i(I_i, H_i) - \sum_{i=0}^{d-1} f_i(I_{i+d}, H_{i+d}) + \sum_{i=0}^{d-1} f_{(d),i}(I_{i+d}, H_{i+d}) \pmod{q} \\ &= \sum_{i=0}^{d-1} f_i(I_i - I_{i+d}, H_i - H_{i+d}) + \sum_{i=0}^{d-1} f_{(d),i}(I_{i+d}, H_{i+d}) \pmod{q} .\end{aligned}$$

Notice that we already know all of the terms in the second summation; let (s, t) be this known vector. If we denote by u the d-dimensional vector with coefficients equal to the first d coefficients of f, then (u, g) is in the following $(N + d + 1) \times (N + d)$ lattice:

$$L_{ug} = \left[\begin{array}{cc} 0 & t \\ \hline I_{(d)} & H_{(N),i} - H_{(N),i+d} \\ 0 & qI_{(N)} \end{array}\right] ,$$

[5] More generally, f is determined by $\{f \pmod{s_1}, f \pmod{s_2}, \ldots, f \pmod{s_z}\}$, $0 \neq s_i \in \mathbb{Z}[X]$, when $(k(X))(X^N - 1) = L.C.M.(s_1, s_2, \ldots, s_z) \pmod{q}$ for some $k(X) \in \mathbb{Z}[X]$.

where $H_{(N),i} - H_{(N),i+d}$ is a $d \times N$ matrix formed by pairing rows, and the top third of the lattice consists only of a single row. Now, if we wish, we may discard the last d columns, obtaining a $(2d+1) \times (2d)$ lattice with target vector (u, v), where v consists of the first d coefficients of g. Clearly, (u, v) is a short vector, most likely the shortest vector in this lattice. Once we obtain (u, v), this information can be combined with $(f_{(d)}, g_{(d)})$ to completely recover (f, g). We thereby obtain the private key without ever having to reduce a $2N$-dimensional lattice. When $N/d > 2$, we can decrease the dimension of L_{CS} by about $2d$ in a similar fashion. This $2d$ reduction in lattice dimension should reduce LLL's running time by a factor exponential in $2d$.

6 NTRU-256

Silverman [13] proposes choosing N to be a power of 2, because then convolution products can be computed rapidly using Fast Fourier Transforms. In particular, he suggests $(N, p, q) = (256, 2, 127)$ as an advantageous choice of parameters. We found that an NTRU-256 private key can be recovered in about 3 minutes using the folding technique described above.

In our experiments, we used a three-staged approach in recovering the private key. First, we recovered $(f_{(64)}, g_{(64)})$ by reducing the lattice generated by $L_{(64)}$ - a 128-dimensional matrix with $H_{(64)}$ in upper right quadrant. Second, we recovered $(f_{(128)}, g_{(128)})$ by reducing the 129×128 lattice constructed as described above. Finally, we took advantage of the modulo p structure of the private keys to create an over-defined system of linear equations. For example, upon computing that $f_{(128),i} = 0$, we know that $f_i = f_{i+128} = 0$, because $f_i \in [0, 1]$ for all i. Similarly, $f_{(128),i} = 2$ implies $f_i = f_{i+128} = 1$. When $p = 2$, this trick most likely results in more than half of g's coefficients being known, and less than half of f's coefficients being unknown, so that we may solve for the unknown coefficients in f. We thereby recover the entire private key (f, g) using lattices one-fourth the size of L_{CS}.

Since the S_f and S_g parameters were not specified for NTRU-(256,2,127), we used those for NTRU-(263,2,127) - specifically, f and g both have 35 1's, the rest 0's [13]. Using the NTL's implementation of LLL with a block size of 10 for both reductions, the 3 stages took an average of 40, 43 and 3 seconds, respectively.[6] Out of 20 trials, the correct key was recovered every time. We also tested the case when f has 75 1's and g has 65 1's, which is more challenging cryptanalytically, both in terms of the lattice reduction (since the target vector is longer) and the linear system (since there are more equations to solve). To avoid errors, the block size for the second reduction was increased to 12. The

[6] For these particular values of S_f and S_g, we could even have begun by recovering $(f_{(32)}, g_{(32)})$, which, heuristically, is recoverable even though N/d is somewhat large. After finding $(f_{(64)}, g_{(64)})$, it is probable that at least half of the coefficients of each of $f_{(64)}$ and $g_{(64)}$ are zero, and that, consequently, over half of the coefficients of each of f and g are known to be zero. If such is the case, we can proceed directly to the third stage without having to reduce a lattice larger than 65×64.

three stages took an average of 84, 94 and 12 seconds, respectively. Out of 10 trials, the correct key was recovered 9 times.

Some gimmicks were required to pick the target vector from among other short, but useless, vectors. For example, in the first reduction, we searched for rather than $(f^{\perp}_{(64)}, g^{\perp}_{(64)})$ rather than $(f_{(64)}, g_{(64)})$, where $f^{\perp}_{(64)}$ is the projection of f orthogonal to 1^N, the vector having every coefficient equal to 1. This technique, originated by Coppersmith and Shamir [1], prevents LLL from returning the short, but cryptanalytically useless vector $(1^N, 0^N)$. Also in the first reduction, LLL would often return other trivial vectors. For example, $(1^{\pm}, 1^{\pm})$ is a trivial vector that can arise in $L_{(64)}$, where $1^{\pm}$ is the vector consisting of alternating 1's and -1's. We simply skipped over these trivial vectors manually, continuing on to the next vector in the lattice until the desired nontrivial one was obtained. This process certainly could have been automated.

7 Remarks on NTRU-Prime

As we noted in the introduction, the domain parameter N is typically chosen to be prime not, apparently, for security reasons, but because it maximizes the probability that a randomly chosen private key f has inverses modulo q and p, these inverses being necessary for public key generation and decryption, respectively [11]. In terms of security, NTRU's typical use of a prime domain parameter appears to be merely fortuitous.

Folding does not work when N is prime — e.g., $f_{(d)} * h_{(d)}$ does not equal $g_{(d)}$ when d does not divide N. However, we can say that if $f * h = g$, and if h has a period of c in the sense that $h_i = h_{i+c}$ for $0 = i < N - c$, then the first N coefficients of $g' = (f, 0) * h'$ (where $(f, 0)$ is f followed by c zeros and h' is h followed by the c coefficients of h's period) are precisely the N coefficients of g. (Proof omitted.) For example, suppose $N = 263$ and h has a period of 37 — i.e., $h_0 = h_{37}, \ldots, h_{225} = h_{262}$. Then, we obtain $(f, 0)$ by appending 37 zeros to f and obtain the last c coefficients of h' by the relations $h_{263} = h_{226}, \ldots, h_{289} = h_{262}$. This will give us $(f, 0) * h' = (g, w)$, where the coefficients of w are not necessarily small. Since $(f, 0)$, h' and (g, w) have dimension $263 + 37 = 300$, a composite number, we can fold them to dimension, say, 100, obtaining the relation $(f, 0)_{(100)} * h'_{(100)} = (g, w)_{(100)}$. All of the coefficients of $(f, 0)_{(100)}$ will be small, and 100 - 37 = 63 of the coefficients of $(g, w)_{(100)}$ will be small, so we can construct a lattice of dimension 163 that may give us partial information about f and g. Although this approach works well in the rare case that h has a small period, it does not appear to lead to an attack against NTRU-prime that works in general.

Circulant lattices have a rather interesting property - namely, given an n-dimensional lattice generated by circulant matrix C, one can construct an $\lfloor \frac{n+1}{2} \rfloor$-dimensional lattice C_{half} that contains a C-vector no more than twice as long as the shortest vector in C. This is a consequence of the fact that for any vectors b and c, $\|b * c\| = \|b_{rev} * c\|$, where b_{rev} is the reverse of b given by $b_{rev,i} = b_{n-i}$. Now, if we suppose that the matrix C consists of cyclic rotations of the vector

c, and that the shortest vector in the lattice generated by C is obtained through left-multiplication by b — i.e., the shortest vector is $b * c$ — we get

$$\|(b + b_{rev}) * c\| \leq \|b * c\| + \|b_{rev} * c\| = 2\|b * c\|.$$

Since $(b + b_{rev})$ is a palindrome — i.e., $(b + b_{rev})_i = (b + b_{rev})_{n-i}$ for all i — $(b + b_{rev})$ has at most $\lfloor \frac{n+1}{2} \rfloor$ distinct coefficients, so that the rows of C can be paired together. This property of circulant lattices does not appear to allow one to cut the dimension of L_{CS} (a *block*-circulant lattice) in half, unless h is a palindrome.

8 Summary and Conclusion

We have shown that choosing N to be a composite number, especially one with a small factor, significantly reduces the security of the NTRU cryptosystem. Also, we have shown that it is possible to recover entire NTRU private keys using lattices of much smaller dimension than was previously thought. To avoid the presented attacks, N should be chosen to be prime, or to have only large nontrivial factors.

Acknowledgements

The author would like to thank Yiqun Lisa Yin, Satomi Okazaki and Atsushi Takeshita for their numerous helpful comments and suggestions.

References

1. D. Coppersmith and A. Shamir. Lattice Attacks on NTRU. In Proc. of Eurocrypt '97, volume 1233 of LNCS, pages 52-61. Springer-Verlag, 1997.
2. E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In Proc. of Crypto '99, volume 1666 of LNCS, pages 537-554. Springer-Verlag, 1999.
3. J. Hoffstein, D. Lieman, J. Pipher, and J.H. Silverman. NTRU: A Public Key Cryptosystem. Submission to IEEE P1363 (1999). Available at http://www.manta.ieee.org/groups/1363/StudyGroup/NewFam.html.
4. J. Hoffstein, J. Pipher, and J.H. Silverman. NTRU: A Ring Based Public Key Cryptosystem. In Proc. of ANTS III, volume 1423 of LNCS, pages 267-288. Springer-Verlag, 1998. Available at http://www.ntru.com.
5. E. Jaulmes and A. Joux. A Chosen-Ciphertext Attack against NTRU. In Proc. of Crypto '00, volume 1880 of LNCS, pages 20-35. Springer-Verlag, 2000.
6. A. May. Cryptanalysis of NTRU. Preprint, February 1999. Available at http://www.informatik.uni-frankfurt.de/~alex/crypto.html.
7. NTRU Cryptosystems. The NTRU Public Key Cryptosystem. Available at http://www.ntru.com/technology/tutorials/pkcstutorial.htm.
8. V. Shoup. Number Theory C++ Library (NTL) version 3.9. Available at http://www.shoup.net/ntl.

9. J.H. Silverman. Dimension-Reduced Lattices, Zero-Forced Lattices, and the NTRU Public Key Cryptosystem. NTRU Cryptosystems Technical Report No.13 (1999). Available at http://www.ntru.com.
10. J.H. Silverman. Estimated Breaking Times for NTRU Lattices. NTRU Cryptosystems Technical Report No.12 (1999). Available at http://www.ntru.com.
11. J.H. Silverman. Invertibility in Truncated Polynomial Rings. NTRU Cryptosystems Technical Report No.9 (1999). Available at http://www.ntru.com.
12. See J.H. Silverman. A Meet-in-the Middle Attack on an NTRU Private Key. NTRU Cryptosystems Technical Report No.4 (1997). Available at http://www.ntru.com.
13. J.H. Silverman. Wraps, Gaps, and Lattice Constants. NTRU Cryptosystems Technical Report No.11 (1999). Available at http://www.ntru.com.

A Appendix

A.1 Probability of Very Short (u, v) Pair

Assume we choose h' from $\mathbb{Z}_q^N$ with uniform distribution; what is the expected length of the shortest nonzero pair (f', g'), $f', g' \in \mathbb{Z}_q^N$, that satisfies $f' * h' = g' \pmod q$? We begin with the (admittedly heuristic) observation that the choices for h' essentially partition the set of (u, v) pairs according to whether $u * h' = v \pmod q$. In other words, it is rare that the set of (u, v) pairs for h'_1 and those for h'_2 overlap such that the equalities $u' * h'_1 = v' \pmod q$ and $u' * h'_2 = v' \pmod q$ are simultaneously satisfied for some (u', v'). This notion could be made more precise, but we will make do with the heuristic observation.

Assuming that the (u, v) pairs are, in fact, partitioned among the choices for h', the probability that a randomly chosen h' has a vector of length less than R is less than or equal to $V(R)/q^N$, where $V(R)$ is the volume of a $2N$-dimensional ball of radius R, and q^N is the number of (u, v) pairs that belong to h'. Using Stirling's Formula for the volume of an n-dimensional ball, we find that the probability that h' has a (u, v) pair shorter than $\sqrt{Nq/2\pi e}$ is negligibly small.

Since the probability of a random h' having a (u, v) pair as short as (f, g) is extremely small, we have some basis for concluding that it is very unlikely that h has a (u, v) pair unrelated to (f, g) that is as small as (f, g). This conclusion comes with caveats, the most important being that the trivial vector $(1^N, 0^N)$, where 1^N is the vector having all N elements equal to 1, is typically a (u, v) pair for h, and may very well be shorter than (f, g). This is not a serious problem, because, as shown in Appendix A.2, the group of (u, v) pairs can be slightly modified to exclude this trivial vector.

A.2 Length of $f_{(d)}$

We can establish an upper bound on the Euclidean norm of $(f_{(d)}, g_{(d)})$ as follows:

Theorem 2. $\|(f_{(d)}, g_{(d)})\| \leq \sqrt{N/d}\|(f, g)\|$.

Proof. Let $b = (1 + X^d + \cdots + X^{(N/d-1)d})$, $v_f = f * b$ and $v_g = g * b$. Since $\|(f * X^i, g * X^i)\| = \|(f, g)\|$ for all i, we have

$$\|(v_f, v_g)\| \leq N/d\|(f, g)\|$$

by the triangle equality, with equality holding when $f = f * X^d = \cdots = f * X^{(N/d-1)d}$ and $g = g * X^d = \cdots = g * X^{(N/d-1)d}$. Notice that the ith coefficient of v_f is equal to $f_{(d),i \bmod d)}$. In other words, the coefficients of v_f are precisely the coefficients of $f_{(d)}$, repeated N/d times. The same goes for v_g. Thus,

$$\|(v_f, v_g)\| = \sqrt{N/d}\|(f_{(d)}, g_{(d)})\| \ ,$$

from which the desired inequality follows. □

For the expected length of $(f_{(d)}, g_{(d)})$, recall that we have $\|(f_{(d)}, g_{(d)})\| = \sqrt{N/d}\|(f, g)\|$ only when

$$f = f * X^d = \cdots = f * X^{(N/d-1)d}, g = g * X^d = \cdots = g * X^{(N/d-1)d}$$

— i.e., when the coefficients of f and g have a period of d. Of course, parameters S_f and S_g can be chosen to require f and g to be periodic, or nearly periodic, but this would reduce the keyspace and invite other attacks.

We can use ideas of Coppersmith and Shamir to obtain a better approximation of the length of the target vector of $L_{(d)}$ when f and g behave like random vectors. Let $f^\perp$ denote the projection of f orthogonal to 1^N, the vector in which all N elements are equal to 1. We find that $v_f^\perp$ — i.e., the projection of v_f orthogonal to 1^N — is equal to $f^\perp * b$. Then, following Coppersmith and Shamir, we get:

$$\begin{aligned}
\|v_f^\perp\|^2 &= \sum_k (f^\perp * b)_k^2 \\
&= \left(\sum_i (f_i^\perp)^2\right)\left(\sum_l b_l^2\right) + \sum_{j\neq 0}\left(\sum_i f_i^\perp f_{i+j}^\perp\right)\left(\sum_l b_l b_{l+j}\right) \\
&= (N/d)\|f^\perp\|^2 + \sum_{j\neq 0}\left(\sum_i f_i^\perp f_{i+j}^\perp\right)\left(\sum_l b_l b_{l+j}\right) \ .
\end{aligned}$$

Since b_l is nonzero only when $l = 0 \pmod d$, each term $b_l b_{l+j}$ must be zero, and thus the entire rightmost summation must be zero, unless $j = 0 \pmod d$. When $j = 0 \pmod d$, the rightmost summation is equal to N/d. Thus, we obtain:

$$\|v_f^\perp\|^2 = (N/d)\|f^\perp\|^2 + \sum_{j\neq 0, j=0 \bmod d} (N/d)\left(\sum_i f_i^\perp f_{i+j}^\perp\right) \ .$$

If f behaves like a random vector, then, for each j, we would expect $\sum_i f_i^\perp f_{i+j}^\perp$ to be less than $\sum_i f_i^\perp f_i^\perp = \|f^\perp\|^2$ by a factor of about $1/\sqrt{N}$. Since the terms

have random sign, we would also expect some cancellation to occur. Thus, if N/d is not too large (there are $N/d - 1$ terms in the first summation), such as when $d > \sqrt{N}$, then we can expect that

$$\|(v_f^{\perp}, v_g^{\perp})\| \approx \sqrt{N/d}\|(f^{\perp}, g^{\perp})\| \ .$$

Since, as before, the coefficients of $v_f^{\perp}$ are precisely the coefficients of $f_{(d)}^{\perp}$ repeated N/d times, where $f_{(d)}^{\perp}$ denotes the projection of $f_{(d)}$ orthogonal to 1^d, we get:

$$\|(f_{(d)}^{\perp}, g_{(d)}^{\perp})\| \approx \|(f^{\perp}, g^{\perp})\| \ .$$

Coppersmith and Shamir have shown that $(f^{\perp}, g^{\perp})$ is the optimal target vector for L_{CS} (ignoring their additional "balancing constant" refinement"). Similarly, $(f_{(d)}^{\perp}, g_{(d)}^{\perp})$ is the optimal target vector for $L_{(d)}$. Thus, when N/d is not too large, we can expect the target vector of $L_{(d)}$ to be about the same length as the target vector of L_{CS}.

Applying the techniques used in Appendix A.1, we find that that while $\|(f_{(d)}^{\perp}, g_{(d)}^{\perp})\| \approx \|(f^{\perp}, g^{\perp})\|$, the expected length of the shortest vector in $L_{(d)}$ is less than that of L_{CS} by a factor of $\sqrt{N/d}$. One might think that this tightening of the ratio between the expected length of the shortest vector to the length of the target vector would make the target vector more difficult for LLL to find, but, empirically, the small reduction in this ratio does not even come close to offsetting the exponential reduction in running times obtained by decreasing the lattice dimension.