NTRU Cryptosystems Technical Report

Report # 013, Version 1

Title: Dimension-Reduced Lattices, Zero-Forced Lattices, and the NTRU Public Key Cryptosystem

Author: Joseph H. Silverman

Release Date: March 9, 1999

Abstract. In this note we describe, extend, and analyze the lattice construction ideas of Alexander May [1] as they apply to the NTRU public key cryptosystem. We use both theoretical and experimental methods to analyze the strength of the attacks. The final conclusion is that the new attacks only marginally affect the security levels of the standard commercial NTRU parameter sets ($N = 167$, 263, and 503), but that the new lattices can be helpful for very low security levels ($N = 107$).

In this note we describe, extend, and analyze the lattice construction ideas of Alexander May [1] as they apply to the NTRU public key cryptosystem. We use both theoretical and experimental methods to analyze the strength of the attacks. The final conclusion is that the new attacks only marginally affect the security levels of the standard commercial NTRU parameter sets ($N = 167$, 263, and 503), but that the new lattices can be helpful for very low security levels ($N = 107$). We will concentrate entirely on the underlying lattices. For details of the NTRU public key cryptosystem, see [2].

§1. The Standard NTRU Lattice.

For the convenience of the reader, we briefly review the set-up of the *Standard NTRU Lattice* L^{NT}. Further details may be found in [3], which also contains the definitions of the various lattice constants referred to below.

Fix integers N, d_f, and d_g. Let S_d be the set of N-tuples with d coordinates equal to each of 1 and -1 and with the remaining $N - 2d$ coordinates equal to 0. Similarly, let S'_d be the set of N-tuples with d coordinates equal to 1, with $d - 1$ coordinates equal to -1, and with the remaining $N - 2d + 1$ coordinates equal to 0. An NTRU private key consists of a pair of vectors

$$f = (f_0, f_1, \ldots, f_{N-1}) \in S'_{d_f} \quad \text{and} \quad g = (g_0, g_1, \ldots, g_{N-1}) \in S_{d_g}.$$

The *Standard NTRU Lattice* L^{NT} is the lattice of dimension $2N$ generated by the row vectors of a matrix of the following form, where $(h_0, \ldots, h_{N-1})$ is a known list of integers:

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

The constant λ is a balancing constant which is chosen to maximize the efficiency of the search for small vectors in the lattice. (It turns out that the best choice for the attacker is $\lambda = \|f\|/\|g\|$.)

The attacker knows that the lattice contains the relatively short vector[†]

$$\mathbf{v} = (\lambda f_0, \ldots, \lambda f_{N-1}, g_0, \ldots, g_{N-1}).$$

Further, it is clear from the cyclical nature of the h portion of the matrix that L^{NT} also contains the vectors obtained by shifting the f and g coordinates of $\mathbf{v}$ cyclically an equal amount. We will denote these shifted vectors by

$$\mathbf{v}^{(k)} = (f_k, f_{k+1}, \ldots, f_{k-1}, g_k, g_{k+1}, \ldots, g_{k-1}).$$

Thus $\mathbf{v} = \mathbf{v}^{(0)}$, and L^{NT} contains all of the vectors $\mathbf{v}^{(0)}, \ldots, \mathbf{v}^{(N-1)}$. The attacker knows the vector h, which is the NTRU public key, so he knows the lattice L^{NT}, and his goal is to recover the unknown vectors f, g from the lattice L^{NT} or some similar lattice.

Extensive experimental evidence [3] (see also [2]) suggests that for lattices in which certain lattice constants are held constant, the log time needed to find a target vector grows (at least) linearly in the dimension. In other words, for families of NTRU-type lattice we have

$$\log(T) \geq A \cdot N + B$$

for certain constants A and B. To give specific examples from [3]

$$\log(T) \geq 0.2002 \cdot N - 7.608 \quad \text{for lattices of type NTRU 167 and NTRU 263,}$$
$$\log(T) \geq 0.1339 \cdot N - 2.9983 \quad \text{for lattices of type NTRU 107.}$$

(The type refers to the value of q and the γ lattice constant described in [3].)

[†] As we have formulated the problem here, the actual NTRU key is the vector $(f_0, f_{N-1}, \ldots, f_1)$, but this reflection of coordinates is irrelevant in studying lattice attacks.

§2. May's Zero-Run Lattices.

Alexander May [1] has introduced what he calls *Zero-Run Lattices* which are designed to take advantage of long runs of 0's in NTRU target vectors. May's basic idea is to multipy r columns of the matrix generating L^{NT} by some large number θ. The effect is to encourage (or even force) the lattice reduction algorithm to find vectors for which those r coordinates are equal to 0, since otherwise those coordinates will have magnitude at least equal to θ, and thus will not be small.

In his paper, May chooses r consecutive columns, which means he is looking for a target vector which contains r consecutive zeros, or as he calls it, a zero-run of length r. Note that it is enough for g to have r consecutive zeros anywhere in its list of coordinates, since then one of the shifted target vectors $\mathbf{v}^{(k)}$ will have r-consecutive zeros in the correct places.

We would like to suggest that rather than multiplying consecutive columns by θ, it is actually advantageous to multiply r random columns of L^{NT} by θ. There are two reasons why using randomly chosen columns is a better strategy:

- It is easy for the key creator to thwart an attack via zero-run lattices by simply requiring that f and g not have long strings of zeros. For example, the key creator could force a ± 1 whenever 10 consecutive zeros have been chosen. If random columns are chosen, this remedy is not available to the key creator.
- The attacker will "win" if the columns he chooses correspond to zero coordinates in one of the shifts $g^{(k)}$ of the the unknown vector g. Now it can happen that there are actually two different shifts $g^{(k_1)}$ and $g^{(k_2)}$ which win. This means that the lattice contains two different target vectors. It turns out that having two target vectors does not seem to help lattice reduction very much. (Indeed, May's underlying idea is to take the lattice L^{NT}, which has N target vectors, and break the symmetry until there is only one target vector.) It turns out that multiple winners are more likely to occur if the attacker chooses consecutive columns than if he chooses randomly chosen columns. Since the total number of winners, counted with multiplicity, in the space of all g's doesn't depend on the choice of columns, and since the attacker's goal is simply to choose a single winning set of columns, he does best if most winners are only single winners. Thus by choosing random columns, he will spread the winning entries as widely as possible.

In conclusion, it is in the attacker's best interest to always choose random columns.

§3. Zero-Forced Lattices.

Regardless of whether the attacker decides to multiply consecutive columns or random columns of L^{NT} by a large number θ, the net effect is to encourage (and ultimately force) the lattice reduction algorithm to find vectors with zeros in those specified coordinates. There is a much more efficient way to achieve the same goal, namely create a smaller dimensional lattice in which the specified coordinates are forced to equal zero. In this section we describe how such *Zero-Forced Lattices* are created. These lattices should always outperform the zero-run lattices described in May's paper. However, we want to stress that these zero-forced lattices are a natural generalization of May's original idea to take advantage of the large number of zeros in the NTRU target vectors.

The first step in forming a Zero-Forced Lattice is to choose a set of indices

$$J = \{j_1, j_2, \ldots, j_r\} \qquad \text{satisfying} \qquad 0 \le j_1 < j_2 < \cdots < j_r < N.$$

We will search for target vectors whose $j_1^{\text{st}}, j_2^{\text{nd}}, \ldots, j_r^{\text{th}}$ coordinates are equal to zero. To do this, we write out the original congruences that were used to form the L^{NT} lattice:

$$f_0 h_j + f_1 h_{j+1} + \cdots + f_{N-1} h_{j-1} \equiv g_j \pmod q, \qquad 0 \le j < N.$$

We are now going to require that $g_{j_1}, \ldots, g_{j_r} = 0$, which gives us r linear relations modulo q for the f_i's.[‡] So we can simply solve these r congruences for $f_{N-r}, \ldots, f_{N-1}$ in terms of $f_0, \ldots, f_{N-r-1}$ and substitute back into the remaining $N - r$ congruences to get a new system of congruences

$$a_{0j} f_0 + a_{1j} f_1 + \cdots + a_{N-1-r,j} f_{N-1-r} \equiv g_j \pmod q \qquad \text{for } 0 \le j < N,\ j \notin J.$$

(To prevent that key creator from stacking the deck against us, it might be better to solve for r random f_k's in terms of the others, rather than using the last r coordinates of f.) Here the a_{ij}'s are known quantities, and the f_k's and g_j's are the unknown quantities. Notice that we now have a system of only $N - r$ congruences in $2(N - r)$ unknowns, so we define the *Zero-Forced Lattice* L_J^{ZF} to be the lattice of dimension $2(N - r)$ spanned by the rows of the following matrix:

$$L_J^{\text{ZF}} = \left(\begin{array}{cccc|cccc}
1 & 0 & \cdots & 0 & & & & \\
0 & 1 & \cdots & 0 & & & & \\
\vdots & \vdots & \ddots & \vdots & & a_{ij} & & \\
0 & 0 & \cdots & 1 & & & & \\
\hline
0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q
\end{array}\right)$$

[‡] For simplicity, we assume the attacker forces zeros in g, since in general g has more zeros than f. A similar analysis works if the attacker forces zeros in f, or in both f and g simultaneously.

The ij indices in the upper righthand part of the matrix are over the range $0 \le i < N-r$ and $0 \le j < N$ with $j \notin J$.

To describe the target vectors in L_J^{ZF}, we let

$$\mathbf{v}_J^{(k)} = \left(f_{i+k}\right)_{0 \le i < N-r} \quad \| \quad \left(g_{j+k}\right)_{j \notin J},$$

where $\|$ indicates concatination of the two vectors. In other words, $\mathbf{v}_J^{(k)}$ is equal to $\mathbf{v}^{(k)}$ with the last r coordinates removed from its f-part and with the J-coordinates removed from the g-part. Note that $\mathbf{v}_J^{(k)}$ will be in the lattice L_J^{ZF} if and only if the J-coordinates of $\mathbf{v}^{(k)}$ are all zero, since we've assumed that they were all zero in creating L_J^{ZF}.

The fact that we've reduced the dimension of the search lattice will certainly speed up the search using LLL or similar lattice reduction algorithms. On the other hand, the lattice L_J^{ZF} won't contain any target vectors at all unless we've chosen a J for which some $\mathbf{v}^{(k)}$ has all of its J-coordinates equal to zero. If r is large, the probability of choosing a good J is small; the estimated search time is inversely proporational to this probability. It is thus very important to estimate this probability.

Proposition. *Fix integers N, m, r. Let $\mathbf{a} = (a_0, \dots, a_{N-1})$ be a randomly chosen vector with $N-m$ of its coordinates equal to zero and the remaining m coordinates non-zero, and let $J = \{j_1, j_2, \dots, j_r\}$ be a set of randomly chosen indices with $0 \le j_1 < j_2 < \cdots < j_r < N$.*

(a) $$\operatorname{Prob}(a_{j_1} = \cdots = a_{j_r} = 0) = \frac{\binom{N-r}{m}}{\binom{N}{m}} = \prod_{i=0}^{m-1} \left(1 - \frac{r}{N-i}\right).$$

(b) $$\operatorname{Prob}\begin{pmatrix} a_{j_1+k} = \cdots = a_{j_r+k} = 0 \\ \text{for some } 0 \le k < N \end{pmatrix} \approx 1 - \left(1 - \prod_{i=0}^{m-1} \left(1 - \frac{r}{N-i}\right)\right)^N.$$

Proof. (a) If we specify that a chosen set of r of the coordinates of $\mathbf{a}$ must be 0, then the number of possible $\mathbf{a}$'s is simply the number of ways to choose m coordinates to be non-zero among the remaining $N-r$ coordinates. There are thus $\binom{N-r}{m}$ ways to choose $\mathbf{a}$ subject to the conditions $a_{j_1} = \cdots = a_{j_r} = 0$. Since there are $\binom{N}{m}$ possible $\mathbf{a}$'s if we impose no conditions, the result follows.
(b) We write $\mathbf{a}^{(k)}$ for $\mathbf{a}$ with its coordinates shifted k-places to the left. From (a) we know the probability of "winning" for each individual $\mathbf{a}^{(k)}$. Hence the probability that some $\mathbf{a}^{(k)}$ wins is simply 1 minus the probability that they all "lose", which is the formula given in (b). [Remark. The reason the formula in (b) is only an approximation is due to the fact that the probability of the different $\mathbf{a}^{(k)}$'s winning are not entirely independent. However, for even moderate values of r and random choices of J, the formula in (b) will be quite accurate. And if one chooses some special J, for example $J = \{0, 1, \dots, r-1\}$ as suggested by May [1], then the actual probability in (b) will be considerably lower than given by the formula.]

Table 1 uses the Proposition to compute the probability that a randomly chosen zero-forced lattice L_J^{ZF} will contain at least one target vector for four sets of sample NTRU parameters. Note that in applying the Proposition, we take $m = 2d_g$, since g contains $2d_g$ non-zero coordinates.

N	d_g	$r = 20$	$r = 30$	$r = 50$
107	12	30.688%	1.162%	0.000%
167	20	37.617%	1.729%	0.001%
263	24	98.112%	34.070%	0.329%
503	72	39.727%	1.400%	0.001%

Table 1. Probability that L_J^{NT} contains a target vector

Table 1 clearly indicates that as r increases, there is a significant loss of efficiency due to the fact that most lattices will have no target vectors at all. Of course, there is also a gain in search efficiency due to the fact that the lattices will have smaller dimension.

For example, if the probability that L_J^{ZF} has a valid target vector is 1%, then one expects to examine roughly 50 lattices before finding one that works. So if the improvement in breaking time due to the lower dimension is less than 50, then the net effect is a loss.

As explained in Section 1 (and in [3]), the breaking time for an NTRU-type lattice grows (at least) like

$$\log(T) = AN + B.$$

A standard NTRU lattice L^{NT} has dimension $2N$, and an associated zero-forced lattice L^{ZF} has dimension $2N - 2r$, so the time gained using the zero-forced lattice instead of the standard lattice is

$$\frac{\exp(AN + B)}{\exp(A(N - r) + B)} = e^{Ar}.$$

In other words, if it takes time T to find a target vector in L^{NT}, then it will take time $e^{-Ar}T$ to find a target vector in L^{ZF}. This looks like a significant gain, but remember that it must be balanced against the fact that if r is large, then it is very unlikely for L^{ZF} to contain any target vectors at all.

Thus the actual gain in time, which we will denote by $\mathrm{gain}(L^{\mathrm{ZF}} : L^{\mathrm{NT}})$, is the product of the probability that L^{ZF} is a winning lattice (i.e., that it has at least one target vector) multiplied by the time gained once we are lucky enough to find a winning lattice. Thus

$$\begin{aligned}\operatorname{gain}(L^{\mathrm{ZF}} : L^{\mathrm{NT}}) &= \operatorname{Prob}\begin{pmatrix} L^{\mathrm{ZF}} \text{ is a} \\ \text{winning lattice}\end{pmatrix} \cdot \left(\frac{\text{Time to find target in } L^{\mathrm{NT}}}{\text{Time to find target in } L^{\mathrm{ZF}}}\right) \\ &= \left(1 - \left(1 - \prod_{i=0}^{2d_g-1}\left(1 - \frac{r}{N-i}\right)\right)^N\right) e^{Ar}.\end{aligned}$$

(Note that $m = 2d_g$ in the proposition, because g has $2d_g$ non-zero coefficients, half of them $+1$ and half of them -1.) The optimal choice of r depends on the original parameter set, specifically on N and d_g, and on the slope of the line $\log(T) = AN+B$ describing how the breaking time increases with dimension. It is difficult to give a formula for this optimal r, but for any particular values N, d_g, A, it is easy to substitute $r = 1, 2, 3, \ldots, 2d_g$ and find the best value.

We now do this for the sample NTRU parameter sets listed in Table 3. We begin with the NTRU 167 and NTRU 263 sets. The experiments in [3] suggest that the breaking time T for these parameter values is

$$\log(T) \approx 0.2002 \cdot N - 7.608.$$

Using $A = 0.2002$ in the formula for the gain, we find the optimal value of r is $r = 18$ for NTRU 167 and $r = 32$ for NTRU 263. Using the same formula for NTRU 503 (see the remark below) gives $r = 18$ optimal. Finally, the extrapolation line for NTRU 107 in [3] is

$$\log(T) \approx 0.1339N - 2.9983,$$

which leads to an optimal $r = 16$. We collect all of this information in Table 2, which also gives the gain, the probability that a randomly chosen L^{ZF} will be a winning lattice, and the expected time T (in MIPS-years) to find and break a winning zero-forced lattice. (The corresponding times for standard NTRU lattices are given in [3, Table 3 and Section 3].)

Parameters	N	d_g	A	r	Prob	gain	T(MIPS-yrs)
NTRU 107	107	12	0.1339	16	0.72	6.16	0.172
NTRU 167	167	20	0.2002	18	0.59	21.57	$9.63 \cdot 10^4$
NTRU 263	263	24	0.2002	32	0.23	139.7	$3.30 \cdot 10^{12}$
NTRU 503	503	72	0.2002	18	0.64	23.56	$1.43 \cdot 10^{34}$

Table 2. Time Gain Using Zero-Forced Lattices

We now make a number of important observations regarding the information contained in Table 2.

N	$\gamma(L)$	q	d_f	d_g	d_ϕ
107	2.658	64	15	12	5
167	3.049	128	61	20	18
263	3.032	128	50	24	16
503	4.081	256	216	72	55

Table 3. Sample NTRU Parameter Sets

Remark. The optimal choice of r and amount of time gained depends very strongly on the slope A of the line relating log breaking time to dimension. If we take a larger value for A, then r and the gain will increase. For example, it is explained in [3] that the slope $A = 0.2002$ for NTRU 167 and NTRU 263 is probably too low, because the graph of the experimental data seems concave up. We used this probably low value for A in [3], because it meant that we were being very conservative in estimating breaking times. Suppose that instead we use the extrapolation line

$$\log(T) \approx 0.2582 \cdot N - 12.484$$

obtained in [3] by dropping the first few data points. Then the optimal values of r for NTRU 167 and NTRU 263 are $r = 21$ and $r = 53$ respectively, and the corresponding gains are 65.8 and 1336. The latter especially seems quite impressive. However, it is an illusion. The reason is that while the zero-forced versus standard gain for NTRU 263 has increased 10-fold from 139.7 to 1336, the actual time needed to break NTRU 263 using zero-forced lattices has increased 3000-fold. The following two statements help to illustrate this point:

- If the correct slope for NTRU 263 is $A = 0.2002$, then it takes $3.65 \cdot 10^{19}$ seconds to break L^{NT}, and it take $2.61 \cdot 10^{17}$ seconds to find and break a winning L^{ZF}, so the gain in time is 139.7.
- If the correct slope for NTRU 263 is $A = 0.2582$, then it takes $1.17 \cdot 10^{24}$ seconds to break L^{NT}, and it take $8.79 \cdot 10^{20}$ seconds to find and break a winning L^{ZF}, so the gain in time is 1336.

The conclusion is that using a smaller slope for extrapolation purposes is always the more conservative approach when estimating breaking times, even though larger slopes increase the relative effectiveness of using zero-forced lattices.

Remark. The previous remark applies especially to our use of the extrapolation line $\log(T) \approx 0.2002 \cdot N - 7.608$ for the NTRU 503 parameter set. As explained in [3], the true extrapolation line for NTRU 503 is almost certainly considerably steeper than this, so the optimal r for NTRU 503 is undoubtedly larger than 18, and the gain from using zero-forced lattice much better than 23.56. However, the net effect of using a more accurate (higher) slope will be that the estimated time needed to break NTRU 503, even using zero-forced lattices, will increase above the already huge value listed in Table 2.

Remark. The experiments in [3] were conducted on families of lattices for which a certain quantity called the γ-constant was held constant. The γ-constant for the standard NTRU lattice is equal to

$$\gamma(L^{\mathrm{NT}}) = \sqrt{\frac{4\pi e \|f\| \cdot \|g\|}{q}}.$$

(See [3] for the definition of γ; for our purposes it suffices to know that as γ increases, it becomes more difficult for lattice reduction algorithms to find the target vector.) From this formula we see that a zero-forced lattice L^{ZF} has the same γ-constant as the standard lattice from which it was created This is true because only zero coordinates have been eliminated from f and/or g, so the quantities $\|f\|$ and $\|g\|$ remain the same. Thus the formula in [3] describing the breaking time for an L^{NT} lattice will also describe the breaking time for its associated L^{ZF} lattices.

Remark. Alexander May (private communication) has pointed out, and we had already noticed, that the use of zero-forced lattices allows at least a partial parallelization of the attack on NTRU. All known lattice reduction methods are sequential in nature, so use of many widely dispersed low power computers is of limited value in speeding up the LLL algorithm. This is in marked contrast to, say, the number field sieve, Pollard's rho method, or the index calculus, which have a probabilistic element making them suitable for distributed computing.

Using zero-forced lattices introduces a similar probabilistic component into lattice reduction. Taking a large r makes it quite unlikely that any particular L^{ZF} will have a winning vector; but if one has access to M separate computers, then each computer can set a different collection of coordinates equal to 0, and it suffices if any one computer chooses a winning lattice. Note, however, that if two or more computers choose winning lattices, no time is gained, since each winning computer will take about the same amount of time to actually find the target vector.

Based on this discussion, it is easy to quantify the time gained through the use of M distributed computers, each of which chooses a zero-forced lattice containing r zeros.

$$\begin{aligned}
\mathrm{Prob}\begin{pmatrix}\text{Some computer has}\\ \text{a winning } L^{\mathrm{ZF}}\end{pmatrix} &= 1 - \mathrm{Prob}\begin{pmatrix}\text{All } M \text{ computers}\\ \text{have losing } L^{\mathrm{ZF}}\text{'s}\end{pmatrix}\\
&= 1 - \mathrm{Prob}\begin{pmatrix}\text{A single computer}\\ \text{has a losing } L^{\mathrm{ZF}}\end{pmatrix}^{M}\\
&= 1 - \left(1 - \prod_{i=0}^{2d_g-1}\left(1 - \frac{r}{N-i}\right)\right)^{NM}.
\end{aligned}$$

To ease notation, we let

$$\beta = \beta(N, d_g, r) = \left(1 - \prod_{i=0}^{2d_g-1}\left(1 - \frac{r}{N-i}\right)\right)^N .$$

so the probability of winning using M computers is $1-\beta^M$. Note that the probability of winning if only one computer is available is $1-\beta$, so the use of M distributed computers increases the probability of winning by a factor of

$$\frac{1-\beta^M}{1-\beta} = 1 + \beta + \beta^2 + \cdots + \beta^{M-1}.$$

If M is large, this will generally be considerably smaller than M, because $\beta < 1$.

In general, the gain using zero-forced lattices distributed on M machines is

$$\mathrm{gain}_M(L^{\mathrm{ZF}} : L^{\mathrm{NT}}) = (1-\beta^M)e^{Ar},$$

where A is the slope of the $\log(T)$ versus N regression line. For any given value of M we can compute this gain for the optimal r and compare it to the gain obtained using a single machine. If this were a true parallelization process, the use of M computers should give an M-fold increase in speed, but as Table 4 makes clear, the benefits of additional distributed processors fall off as M increases. The column labeled "efficiency" measure to what extent using M processors leads to an M-fold decrease in breaking time. Thus an efficiency of 25% means that M processors decreased the breaking time by a factor of $M/4$. The data in Table 4 is for NTRU 263 and slope $A = 0.2002$. We have chosen NTRU 263 as our example because its small d_g makes it the most attractive for zero-forced lattice attacks.

M	r	gain_M	Efficiency
1	32	140	100.00%
5	38	549	78.60%
10	40	978	69.98%
25	44	2072	59.31%
100	49	6299	45.08%
500	55	22048	31.56%
1000	57	37444	26.80%
1500	58	50744	24.21%

Table 4. Efficiency of Using M Distributed Processors

§3. Dimension Reduced Lattices.

The idea of using zero-forced (or May's zero-run) lattices is to force certain coordinates of the target vector to be zero. An alternative idea, also due to May and described in his paper [1], is to simply discard some of the coordinates of the target vector. Qualitatively, this has two effects, one good and one bad:

- The dimension of the lattice goes down, which helps the attacker.
- The the length of the target vector gets closer to the smallest expected non-zero length, which hurts the attacker.

We now quantify these effects. We begin with a lattice L generated by the rows of a $2n$-by-$2n$ matrix of the form

$$L = \begin{pmatrix} I_n & A \\ 0 & qI_n \end{pmatrix}.$$

Thus we could take L to equal the usual NTRU lattice L^{NT}, in which case $n = N$; or we might take L to be a zero-forced lattice L_J^{ZF}, in which case $n = N - r$. We write

$$\mathbf{v} = (u, v) \in L$$

for the target vector, where u and v are n-tuples. More generally, if u and v have unequal lengths, we multiply the first n columns of L by a balancing constant λ which is chosen to make the lattice search as efficient as possible. (We will see below that the optimal choice is $\lambda = \|v\|/\|u\|$.)

Next fix constants $0 < \alpha, \beta \le 1$ with $\alpha + \beta \ge 1$. Starting with the matrix of L, we randomly choose βn of the first n columns and αn of the second n columns. We keep these $(\alpha + \beta)N$ columns and discard the other columns. We denote by $L_{\alpha,\beta}$ the lattice generated by the rows of the resulting $2n$-by-$(\alpha + \beta)n$ matrix, and we call it the (α, β)-*dimension reduced lattice* associated to L.

Note that although $L_{\alpha,\beta}$ is generated by $2n$ vectors, it has dimension $(\alpha+\beta)n$, since its elements are $(\alpha + \beta)n$-tuples. The target vector in $L_{\alpha,\beta}$ is the vector

$$\mathbf{v}_{\alpha,\beta} = (\lambda u_\beta, v_\alpha)$$

which has length

$$\|\mathbf{v}_{\alpha,\beta}\| = \sqrt{\lambda^2\|u_\beta\|^2 + \|v_\alpha\|^2} \approx \sqrt{\beta\lambda^2\|u\|^2 + \alpha\|v\|^2}.$$

The matrix defining $L_{\alpha,\beta}$ has the form

$$\begin{pmatrix} \lambda J_{\beta n} & A_\alpha \\ 0 & qJ_{\alpha n} \end{pmatrix},$$

where J_m denotes an n-by-m matrix such that each column has exactly one 1, and each row has either zero or one 1, with all other entries 0. The discriminant of $L_{\alpha,\beta}$ is the greatest common divisor of the $(\alpha+\beta)n$-by-$(\alpha+\beta)n$ subdeterminants of the matrix. To form a non-zero subdeterminant, we certainly need to take the βn rows which have λ's in them. If we also take the other $(1-\beta)n$ rows from the top half, we are still forced to take $\alpha n-(1-\beta)n$ of the rows in the second half which have q's in them. Hence any non-zero subdeterminant will divide

$$\lambda^{\beta n} q^{(\alpha+\beta-1)n},$$

and in general the gcd of the subdeterminants will equal this quantity. Thus we may take $\operatorname{Disc}(L_{\alpha,\beta})=\lambda^{\beta n} q^{(\alpha+\beta-1)n}$.

We compute the various quantities associated to $L_{\alpha,\beta}$. (See [3] for the definitions of these quantities.)

$$\dim(L_{\alpha,\beta})=(\alpha+\beta)n, \qquad \operatorname{Disc}(L_{\alpha,\beta})=\lambda^{\beta n}q^{(\alpha+\beta-1)n}$$

$$\sigma(L_{\alpha,\beta})=\sqrt{\frac{\dim(L_{\alpha,\beta})}{2\pi e}}\operatorname{Disc}(L_{\alpha,\beta})^{1/\dim L_{\alpha,\beta}}=\sqrt{\frac{(\alpha+\beta)n}{2\pi e}}\left(\lambda^{\beta}q^{\alpha+\beta-1}\right)^{1/(\alpha+\beta)}$$

$$\tau(L_{\alpha,\beta})=\|\mathbf{v}_{\alpha,\beta}\|\approx\sqrt{\beta\lambda^2\|u\|^2+\alpha\|v\|^2}$$

The attacker wants to make the ratio $\tau(L_{\alpha,\beta})/\sigma(L_{\alpha,\beta})$ as small as possible, and one easily checks that the best choice is $\lambda=\|v\|/\|u\|$. This leads to a lattice constant

$$\frac{\tau(L_{\alpha,\beta})}{\sigma(L_{\alpha,\beta})}=\frac{1}{q}\sqrt{\frac{2\pi e}{n}}\left(\|u\|^{\beta}\|v\|^{\alpha}q\right)^{1/(\alpha+\beta)}.$$

Assuming that $q\|u\|\geq\|v\|$ and $q\|v\|\geq\|u\|$ (as will certainly be the case for NTRU lattices), it is easy to check that τ/σ is strictly increasing as (α,β) decreases, starting from $(1,1)$. (This is a nice exercise. It's easiest to first take logarithms of both sides.) Table 5 lists a few values for the parameter set

$$N=263, \quad q=128, \quad d_f=50, \quad d_g=24.$$

$\alpha\backslash\beta$	1.0	0.9	0.8	0.7	0.6	0.5
1.0	0.187	0.210	0.240	0.277	0.327	0.395
0.9	0.214	0.244	0.283	0.335	0.405	0.502
0.8	0.249	0.290	0.343	0.414	0.515	0.662
0.7	0.296	0.350	0.425	0.529	0.681	0.914
0.6	0.358	0.435	0.542	0.700	0.942	1.33
0.5	0.446	0.557	0.720	0.971	1.38	2.11

Table 5. $\tau(L_{\alpha,\beta})/\sigma(L_{\alpha,\beta})$ for $N=263$

It is clear from experiments that the time needed to find the target vector increases quite rapidly as the lattice constant τ/σ increases. This is also clear theoretically, since as τ/σ gets closer to 1, the lattice reduction algorithm will have more and more difficulty picking out the target vector from the large number of vectors of length approximately σ. Further, the length of the target vector and of the smallest expected vectors will be getting closer to the length of the q-vectors (note q won't change), and as noted in [2] and [3], this will further decrease the efficiency of lattice reduction methods. More precisely, it will increase the block size needed before the lattice reduction algorithm is able to find any vector smaller than a q-vector. Note that the running time is exponential in the block size.

The experiments performed by May [1] in low dimensions show a modest increase in speed for the choices $(\alpha,\beta)=(0.5,1.0)$ and $(\alpha,\beta)=(0.8,0.8)$. May also finds a somewhat better increase at the highest dimension he can handle, around $N=100$ to $N=107$, but even for these dimensions the speed increase is not substantial enough to seriously affect the security estimates for suggested NTRU parameters $N=167$, $N=263$, and $N=503$. Further, much of the gain at these dimensions seems to reflect the experimentally observed fact that current lattice reduction methods appear to "hit a wall" around dimension 200, at which point running times jump quite substantially. So using May's dimension reducing idea for lattices whose dimension is a little past this wall can reduce the dimension to before the wall, giving (in May's data) up to 10-fold increases in speed. This does not affect the NTRU security analysis, since at the recommended dimensions one will not see a similar speed-up (unless there is another "wall" near the NTRU parameter dimensions, in which case the speed loss due to the wall would far outweigh any gain from minor dimension reduction).

References

[1] Alexander May, Cryptanalysis of NTRU, preprint, February 1999

[2] J. Hoffstein, J. Pipher, J.H. Silverman, NTRU: A new high speed public key cryptosystem, 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

[3] J.H. Silverman, Estimated Breaking Times for NTRU Lattices, NTRU Cryptosystems Technical Report 012, March 2, 1999. ⟨www.ntru.com⟩