NTRU Cryptosystems Technical Report Report # 012, Version 2: Estimated Breaking Times for NTRU Lattices

Jeffrey Hoffstein, Joseph H. Silverman, William Whyte

NTRU Cryptosystems,
5 Burlington Woods,
Burlington, MA 02144

Abstract. In this note we report on experiments with the lattices underlying the NTRUEncrypt and NTRUSignPublic Key Cryptosystems. We present data for the time needed to find a small vector and use this data to extrapolate expected breaking times for NTRUEncrypt for recommended parameter values. We also extend the "zero-forcing" analysis of [7] to include a check that the lattice strength in the lower-dimension, zero-forced lattice can correctly be approximated by the same extrapolation line as the non-zero-forced lattice.

1 Introduction

In this note we report on experiments with the lattices underlying the NTRUEncrypt Public Key Cryptosystem. These experiments extend those described in [1]. We will concentrate entirely on the underlying lattices. For details of NTRUEncrypt, see [1].

2 The Standard NTRU Lattice

Fix integers N, q, d_f, and d_g. For any d, $0 < d < N$, let $S(d)$ be the set of N-tuples with d coordinates equal to 1 and with the remaining $N-d$ coordinates equal to 0. Let the private key f, g be chosen at random from the spaces $S(d_f), S(d_g)$ and let

$$h = f^{-1} * g$$

be the public key[1]. Here, as usual, operations occur in the ring

$$Z[X]/((X^N - 1), q) \ ,$$

and we identify the vector $(a_0, a_1, a_2, \ldots, a_{N-1})$ with the polynomial $a_0 + a_1 X + a_2 X^2 + \cdots + a_{N-1} X^{N-1}$.

[1] For efficiency reasons, h is often transmitted as $p * f^{-1} * g$; this does not affect our analysis.

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 $h = (h_0, \ldots, h_{N-1})$:

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

The constant λ is a balancing constant which is chosen to maximize the efficiency of the search for small vectors in the lattice.

An NTRUEncrypt private key consists of a pair of vectors

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

and the attacker knows that the lattice contains the relatively short vector[2]

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

which we will refer to as the *target vector*.

The attacker knows h, which is the NTRUEncrypt public key, so he knows the lattice L^{NT}, and his goal is to recover the unknown vector $\mathbf{v}$, or any other vector of approximately the same length, from L^{NT}. (For example, L^{NT} contains N vectors of the same length as $\mathbf{v}$ obtained by cyclically rotating the coordinates of f and g.)

The length of a vector

$$\mathbf{w} = (\lambda w_0, \ldots, \lambda w_{N-1}, w'_0, \ldots, w'_{N-1})$$

is defined to be the centered norm

$$\|\mathbf{w}\| = (\lambda^2(w_0 - \mu)^2 + \cdots + \lambda^2(w_{N-1} - \mu)^2 + (w'_0 - \mu')^2 + \cdots + (w'_{N-1} - \mu')^2)^{1/2},$$

where

$$\mu = (w_0 + \cdots + w_{N-1})/N \quad \text{and} \quad \mu' = (w'_0 + \cdots + w'_{N-1})/N.$$

Srictly speaking, the optimal lattice problem to be solved to locate the target vector $\mathbf{v}$ is a closest vector problem. Specifically, $\mathbf{v} \in L^{\mathrm{NT}}$ will be a distance $\|\mathbf{v}\|$ from the non-lattice vector

$$(\lambda\mu, \ldots, \lambda\mu, \mu', \ldots, \mu')$$

[2] As we have formulated the problem here, the precise NTRUEncrypt key is the vector $(f_0, f_{N-1}, \ldots, f_2, f_1)$, but this rearrangement of coordinates is irrelevant in studying lattice attacks.

and this distance will be a bit shorter than the usual L^2 norm of $\mathbf{v}$. As a practical consequence this means that one will generally add an extra row and column to L^{NT} and try to solve a closest vector problem in this new lattice where the dimension is increased by 1. In this exposition we will ignore this technicality and do our analysis in the lattice L^{NT}.

Two constants seem to be very relevant to the problem of locating short vectors in lattices of the form L^{NT} using LLL type techniques. These are

$$\begin{aligned} a &= N/q, \\ c &= \sqrt{4\pi e\|f\|\|g\|/q}. \end{aligned}$$

These are related to basic constants associated to L^{NT} in the following way. Let

$$\begin{aligned} \sigma &= \text{shortest expected (non-zero) length of vectors in } L^{\mathrm{NT}}, \\ \|\mathbf{v}\| &= \text{length of actual shortest vector in } L^{\mathrm{NT}}. \end{aligned}$$

For a general lattice L, the Gaussian heuristic says that the length of the shortest non-zero vector satisfies

$$\sigma(L) \approx \sqrt{\frac{\dim(L)}{2\pi e}}\,\mathrm{Det}(L)^{1/\dim(L)}.$$

The dimension and determinant of L^{NT} are given by

$$\dim(L^{\mathrm{NT}}) = 2N, \qquad \mathrm{Det}(L^{\mathrm{NT}}) = (\lambda q)^N$$

and so in L^{NT} we have

$$\sigma = \sqrt{\frac{Nq\lambda}{\pi e}}$$

and

$$\|\mathbf{v}\| = \sqrt{\lambda^2\|f\|^2 + \|g\|^2}\,.$$

For optimal use of lattice reduction techniques the balancing constant λ should be chosen to make the ratio $\|\mathbf{v}\|/\sigma$ as small as possible, since lattice reduction methods have been widely observed to work best when the target vector is as small as possible compared to the many vectors of length approximately σ.

It is easily checked that the optimal choice for the balancing constant is $\lambda = \|g\|/\|f\|$, which leads to

$$\sigma = \sqrt{\frac{Nq\|g\|}{\pi e\|f\|}}.$$

Now $\|\mathbf{v}\|^2 = \lambda^2\|f\|^2 + \|g\|^2 = 2\|g\|^2$ and so $\mathbf{v}, c$ and σ satisfy the relation

$$c = \sqrt{2N}\frac{\|\mathbf{v}\|}{\sigma}\,.$$

A large number of experiments that we have performed over the past 6 years support the hypothesis that if one holds c and a constant, while increasing N then the log time to find the target vector grows at least linearly with N. In other words, if we let $T = T(L^{\mathrm{NT}})$ denote the amount of time it takes for LLL to find a target vector in the lattice L^{NT} of dimension $2N$, then

$$\log T \geq AN + B$$

for constants A and B. (In fact, the graph of $\log T$ against N is consistently convex upwards.) For N even moderately large, i.e. greater than 50, if $\|\mathbf{v}\| < \sigma/2$ then LLL will consistently either fail to find a moderately short vector, or will locate the target $\mathbf{v}$ or one of its rotations. This lends some credence to the use of the Gaussian heuristic for lattices of this form.

The constants A and B depend upon c and a in a way that we have not yet quantified precisely. However it is clear that as either c or a or both increase, the constant A increases. Thus it appears to be true that c, a, N provide a measure of the difficulty of finding the target vector $\mathbf{v}$ in L^{NT}. In particular, as c, a, N increase, the length of time necessary to locate $\mathbf{v}$ increases.

2.1 Alternative NTRU Lattice Problems

The paper [4] suggests taking f to be of the form $1 + pF$, $F \in S(d_F)$. In this case, the relevant lattice problem is derived as follows:

$$\begin{aligned} f * h &= g \bmod q \\ \Rightarrow (1 + pF) * h &= g \bmod q \\ \Rightarrow F * (ph) &= g - h \bmod q \ . \end{aligned}$$

We therefore know that the vector $(F, g - h)$ is in the lattice

$$\begin{pmatrix} 1 & ph \\ 0 & q \end{pmatrix} \ ,$$

and is (F, g) away from the known non-lattice vector $(0, -h)$. The attacker's strategy is therefore to try to solve a CVP in this new lattice. The associated lattice reduction problem will have the same expected running time as the reduction problem in the standard NTRU lattice described above, because the discriminant, the dimension, and the constants c and a are unchanged. Therefore, although the current recommended parameter sets [1] take $f = 1 + pF$, it is sufficient to analyze the running time of the case $f \in S(d_F)$.

3 Experimental Results

All of the experiments in this note were run on 400 MHz Celeron machines running the Linux operating system. The software used was Victor Shoup's implementation of the LLL algorithm with improvements due to Schnorr, Euchner

and Hoerner. Shoup's NTL package is available at [8]. Most of the general remarks in [5] concerning lattice reduction algorithms apply to the experiments in this note. In particular, we set Schnorr's pruning constant to equal 0, because we did not find that setting it to be positive improved the running time. We also set the LLL constant $\delta = 0.99$, and we ran the program using increasing block sizes until it found the target vector.

The experiments described in this note confirm the observation made in [5] that (at least for the NTRU lattices) the algorithm generally either finds a vector of the exact correct length, or it finds one that is considerably too long to be useful for decryption. Thus the idea of Coppersmith and Shamir [2] to use vectors a little longer than the target vector to attack NTRU, while very interesting as a theoretical remark, does not appear to be of practical significance. In practice, LLL generally seems to terminate with no significant progress until a sufficiently large block size is used, at which point it finds the target vector. The necessary block size increases roughly linearly with the dimension, and as one knows, the running time of LLL increases exponentially with the block size.

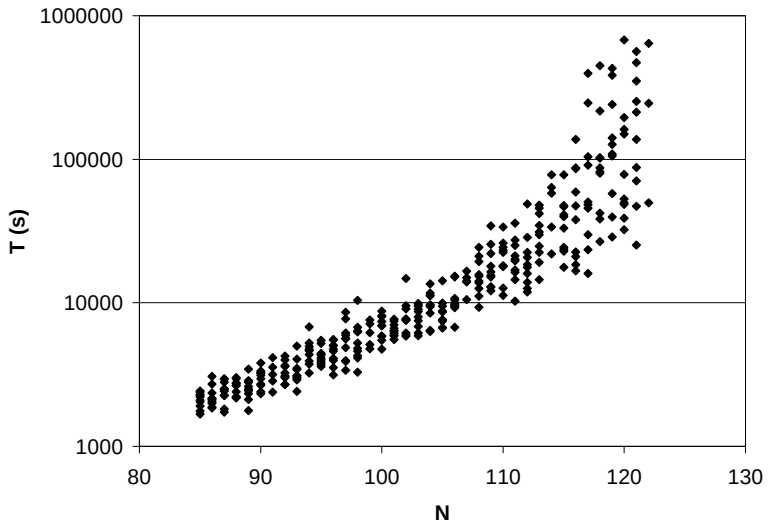

Fig. 1. Results of all experiments on breaking times. Times in seconds.

In general we have found that the smaller the values of c, a the larger the N we can successfully apply LLL to. It was thus interesting to take a particularly small set of values and continue the computation to as large an N as possible.

The smallest example we have investigated so far is the case $c = 1.73, a = 0.535$. Figure 1 gives the results of the experiments run to date. Times in this figure are given in seconds; since the experiments were run on 400 MHz Celeron machines, the time in seconds is converted to the time in MIPS-years by first multipying by 400 (to account for the 400 MHz machines) and then dividing by 31557600, which is the number of seconds in a year. In this case the graph of $\log(T_{\text{avg}})$ versus N is reasonably linear, although there is a noticable upward concavity.

N_{init}	A	B	$T(N = 251)$	$T(N = 503)$
85	0.054	-6.237	1.47×10^{7}	5.78×10^{20}
95	0.066	-7.590	7.15×10^{8}	3.19×10^{25}
100	0.076	-8.714	1.60×10^{10}	2.17×10^{29}
105	0.089	-10.219	9.18×10^{11}	2.36×10^{34}
110	0.104	-12.036	1.06×10^{14}	2.15×10^{40}
115	0.121	-13.965	1.45×10^{16}	3.51×10^{46}

Table 1. Extrapolation line $\log_{10}(T) = AN + B$ depending on the start point. Times are in MIPS-years.

Because of this concavity, the projected breaking time for high values of N depends greatly on the extrapolation method used. Our approach was as follows:

- Select a start point N_{init}.
- For each value of $N \geq N_{\text{init}}$, average the breaking times.
- Plot the linear regression line through the log of these averages, $\log_{10}(T) = AN + B$.

Table 1 shows how the regression line, and the projected breaking times for the typical values of $N = 251$ and $N = 503$, depend on the value selected for N_{init}.

Our choice is to take $N_{\text{init}} = 111$. This allows us to perform the linear extrapolation with 12 values of N and 120 experimental values for T. We computed the linear regression line for $\log_{10}(T)$ and found the values $A = 0.1095, B = -12.6402$ with a correlation coefficient of .9807. The relevant data points are shown in figure 2.

The formula

$$\log_{10} T \geq 0.1095N - 12.6402$$

leads to the extrapolated breaking times given in table 2. Note that the values of N given in table 2 are illustrative: only prime values of N are recommended.

We repeat that the choice of N_{init}, and the use of the linear extrapolation, are somewhat conservative: the trend of the breaking times is clearly concave upwards. It should be borne in mind that there is probably a lot of slack in the figures given in this note, and that additional measurements at $N > 120$ may allow us to improve our estimates of the lattice security of the current parameter sets.

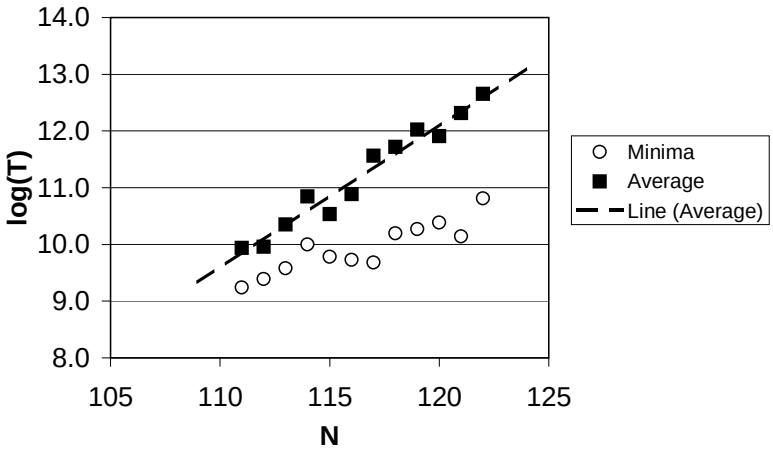

Fig. 2. Average breaking time and minimum breaking time, in seconds, for $N \geq 111$.

N	T	Bit security
167	3.21×10^{5}	57
251	5.07×10^{14}	88
400	1.05×10^{31}	142
500	9.33×10^{41}	178
600	8.31×10^{52}	214
800	6.59×10^{74}	287
1000	5.23×10^{96}	360

Table 2. Breaking times and bit security for various N using the extrapolation line $\log_{10} T \geq 0.1095N - 12.6402$.

The above analysis gives the base level of security against lattice attack. However, actual parameter sets must take into account *zero-forcing* techniques that reduce the effective dimension. This is discussed in the next section.

4 Recommended Parameter Sets: Breaking Times and Zero-Forcing

EESS#1, version 2 [1], suggests the parameter sets `ees251ep4` and `ees251ep5`, both of which have

$$N = 251, q = 239, f = 1 + pF, d_f = 72, d_g = 72, d_r = 72,$$

leading to the lattice values

$$c = 2.709 > 1.73, \quad a = 1.05 > 0.535.$$

However, we cannot simply estimate the security by substituting $N = 251$ into the formula

$$\log_{10} T \geq 0.1095N - 12.6402\,,$$

because this does not take into account the effects of *zero-forcing*, described in [7]. Zero-forcing consists of guessing a pattern of r zeroes that appear in f; if the guess is correct, the effective dimension of the lattice is reduced to $2N - r$. If the log of the breaking time for the lattice without zero-forcing is given by $\log_{10}(T) = AN + B$, then zero-forcing should reduce this log by $Ar/2$. However, this result is only true if the zero-forcing does not significantly change the characteristics of the lattice. We therefore briefly discuss this question.

Change in a due to zero-forcing — The parameter a goes from N/q to $(2N - r)/2q$.

Change in c due to zero-forcing — The centered norm of g in the zero-forced lattice is the same as in the original lattice, neglecting the balancing factor; however, the centered norm of f becomes $\sqrt{d_f(1 - d_f/(N - r))}$. As r increases, this centered norm will decrease, leading to a shorter short vector. The Gaussian length σ in the unbalanced lattice will increase, because the discriminant (the volume of the fundamental parallelopiped) will stay the same but the dimension will decrease. To be precise,

$$\begin{aligned}\sigma(L) &\approx \sqrt{\frac{\dim(L)}{2\pi e}}\,\mathrm{Det}(L)^{1/\dim(L)} \\ &= \sqrt{\frac{2N - r}{2\pi e}} q^{\frac{N}{2N-r}} \lambda^{\frac{N-r}{2N-r}}\,.\end{aligned}$$

We therefore expect the zero-forcing to result in a c more favourable to the attacker. To quantify this effect, we next calculate the appropriate value for the balancing constant λ. This turns out to be

$$\lambda_{\mathrm{zf}} = \frac{\|g\|}{\|f\|_{\mathrm{zf}}}\sqrt{\frac{N - r}{N}}\,,$$

leading to

$$c_{\mathrm{zf}} = \sqrt{\frac{2(2N-r)\pi e}{N}} \left(\frac{\|g\|}{q}\right)^{\frac{N}{2N-r}} \left(\frac{N\|f\|_{\mathrm{zf}}^2}{N-r}\right)^{\frac{N-r}{2(2N-r)}}$$

(note that when $r = 0$ we recover the original value of c).

Applying formula (4) of [7] we see that the optimal choice of r for $N = 251, d = 72, A = 0.1095$ is $r = 17$, leading to a speed up by a factor of 34. Using the formulae above, we find

$$c_{\mathrm{zf}} = 0.901c = 2.442 > 1.73, \quad a_{\mathrm{zf}} = 1.014 > 0.535 \ ,$$

so the extrapolation line with $A = 0.1095$ is still valid. The expected security is therefore:

$$T = 1.37 \times 10^{13} \text{MIPS-years};$$
$$\text{Bit security} = 83 \ .$$

As previously noted, our heuristic result is that higher values of c and a lead to increased breaking times at constant N. We therefore state with some confidence that the time to break the given parameter sets will be at least 10^{12} MIPS-years. The time to break an 80-bit symmetric cryptosystem is typically taken to be about 10^{12} MIPS-years [6]; this estimate of strength thus allows us to recommend the above parameter set for use in systems that require 80-bit security.

5 Further notes

This section outlines areas for future research.

New lattice techniques — It is to be expected that new lattice reduction techniques will be discovered over time, or that currently existing lattice reduction techniques which have not yet been applied to the NTRU lattice will prove to be more efficient than the specific technique used in this technical note. These will gradually reduce the estimated strength of this and other parameter sets. This technical note will continue to be updated with the latest technical results as appropriate.

Improved implementation of current techniques — The experiments in this note were run using NTL [8], which is not optimized for NTRU lattices. Conceivably, simply optimizing for the specific case under consideration could give a speedup by a large constant factor. However, as noted before, the estimates of lattice strength in this note are highly conservative, and a speedup by a constant factor is unlikely to have a practical effect on the security of the system.

NTRU-specific techniques — Currently, very few techniques for lattice reduction are known that take advantage of the special properties of the NTRU lattice. Notable examples of such techniques are [3, 7]. As additional NTRU-specific techniques are discovered and implemented this technical note will be updated appropriately.

References

1. Consortium for Efficient Embedded Security, *Efficient Embedded Security Standard #1*, Version 2, May 2003, available from `http://www.ceesstandards.org`.
2. D. Coppersmith, A. Shamir, Lattice attacks on NTRU, in W. Fumy, ed., *Proceedings fo EUROCRYPT 97*, Lecture Notes in Mathematics **1233**, Springer, 1997, 52–61
3. C. Gentry, Key recovery and message attacks on NTRU-composite, *Advances in Cryptology —Eurocrypt '01*, LNCS 2045. Springer-Verlag, 2001
4. J. Hoffstein and J. H. Silverman. Optimizations for NTRU. In Publickey Cryptography and Computational Number Theory. DeGruyter, 2000.
5. 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
6. A.K. Lenstra, E.R. Verheul, *Selecting Cryptographic Key Sizes*, Journal of Cryptology vol. 14, no. 4, 2001, 255-293.
7. A. May, J.H. Silverman, *Dimension reduction methods for convolution modular lattices*, in Cryptography and Lattices Conference (CaLC 2001), J.H. Silverman (ed.), Lecture Notes in Computer Science 2146, Springer-Verlag, 2001
8. NTL — A Number Theory Library, Victor Shoup, available at `http://www.cs.wisc.edu/~shoup/ntl/`

Comments and questions concerning this technical report should be addressed to `techsupport@ntru.com`

Additional information concerning NTRU Cryptosystems and the NTRU Public Key Cryptosystem are available at `www.ntru.com`