Choosing Parameters for NTRUEncrypt

Jeff Hoffstein[1], Jill Pipher[1], John M. Schanck[2,3],
Joseph H. Silverman[1], William Whyte[3], and Zhenfei Zhang[3]

[1] Brown University, Providence, USA
{jhoff,jpipher,jhs}@math.brown.edu
[2] University of Waterloo, Waterloo, Canada
[3] Security Innovation, Wilmington, USA
{jschanck,wwhyte,zzhang}@securityinnovation.com

Abstract. We describe a methods for generating parameter sets and calculating security estimates for NTRUEncrypt. Analyses are provided for the standardized product-form parameter sets from EESS #1 and for the NTRU Challenge parameter sets.

1 Introduction and Notation

In this note we will assume some familiarity with the details and notation of NTRUEncrypt. The reader desiring further background should consult standard references such as [4, 5, 12]. The key parameters are summarized in Table 1. Each is, implicitly, a function of the security parameter λ.

Primary NTRUEncrypt Parameters	
(N, q)	Ring parameters $R_{N,q} = \mathbb{Z}_q[X]/(X^N - 1)$.
p	Message space modulus.
(d_1, d_2, d_3)	Non-zero coefficient counts for product form polynomial terms.
d_g	Non-zero coefficient count for private key component g.
d_m	Message representative Hamming weight constraint.

Table 1.

NTRUEncrypt uses a *ring of convolution polynomials*; a polynomial ring parameterized by a prime N and an integer q of the form

$$R_{N,q} = (\mathbb{Z}/q\mathbb{Z})[X]/(X^N - 1).$$

The subscript will be dropped when discussing generic properties of such rings. We denote multiplication in R by $*$. An NTRUEncrypt public key is a generator for a cyclic R-module of rank 2, and is denoted $(1, h)$. The private key is an element of this module which is "small" with respect to a given norm and is denoted (f, g). Ring elements are written in the monomial basis; elements of $\mathbb{Z}/q\mathbb{Z}$ are identified with their unique integer representatives in $[-q/2, q/2]$ when lifted to $\mathbb{Z}$ or reduced by a second modulus; and the aforementioned norm is the 2-norm on coefficient vectors:

$$\left\| \sum_{i=0}^{N-1} a_i x^i \right\|^2 = \sum_{i=0}^{N-1} a_i^2. \tag{1}$$

This norm is extended to elements $(a, b) \in R \oplus R$ as

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

There is a large degree of freedom in choosing the structure of the private key. In previous parameter recommendations [12][13] the secret polynomials f and g have been chosen uniformly from a set of binary or

trinary polynomials with a prescribed number of non-zero coefficients. These are far from the only choices. The provably secure variant of NTRUEncrypt by Stehlé and Steinfeld [20], samples f and g from a discrete Gaussian distribution, and the NTRU-like signature scheme BLISS [17] samples its private keys from a set of polynomials with a prescribed number of ± 1s and ± 2s. The reasons for such choices are varied: binary polynomials were believed to allow for a small q parameter, but the desire to increase resistance against the hybrid combinatorial attack of [11] motivated the use of larger sample spaces in both NTRUEncrypt and BLISS. In the provably secure variant the public key must be computationally indistinguishable from an invertible ring element chosen uniformly at random. The discrete Gaussian distribution has several nice analytic properties that simplify the proof of such a claim, and sampling from such a distribution is reasonably efficient.

The parameter choices here make use of product-form polynomials for the private key component f and for the blinding polynomials used during encryption. First introduced to NTRUEncrypt in [9], product form polynomials allow for exceptionally fast multiplication in R without the use of Fourier transforms.

We set the notation:

$$\begin{aligned}\mathcal{T}_N &= \{\text{trinary polynomials}\}\\ \mathcal{T}_N(d,e) &= \left\{\begin{array}{l}\text{trinary polynomials with exactly}\\ d \text{ ones and } e \text{ minus ones}\end{array}\right\}\\ \mathcal{P}_N(d_1,d_2,d_3) &= \left\{\begin{array}{l}\text{product form polynomials}\\ A_1 * A_2 + A_3 : A_i \in \mathcal{T}_N(d_i,d_i)\end{array}\right\}.\end{aligned}$$

If N is fixed, we drop it from the notation and write $\mathcal{T}$, $\mathcal{T}(d,e)$, and $\mathcal{P}(d_1,d_2,d_3)$.

A *product form private key*, the only type that will be considered in this document, is $(f,g) = (1+pF, g)$ with $F \in \mathcal{P}_N(d_1,d_2,d_3)$ and $g \in \mathcal{T}_N(d_g+1, d_g)$. As an additional caveat we need f to be invertible in $R_{N,q}$ so that the corresponding public key $(1,h) = (1, f^{-1}g)$ exists. The parameters recommended in this document ensure that when F is sampled uniformly from $\mathcal{P}_N(d_1,d_2,d_3)$, the polynomial $1+pF$ will always be invertible. One may optionally check that g is invertible, although this is similarly unnecessary for appropriately chosen parameters.

Algorithm 1 NTRUEncrypt Product Form Key Generation

Input: A full set of NTRUEncrypt parameters.

1: **repeat**
2: $\quad F \leftarrow_{\$} \mathcal{P}_N(d_1,d_2,d_3)$
3: $\quad f = 1 + pF \in R_{N,q}$
4: **until** f is invertible in $R_{N,q}$
5: $g \leftarrow_{\$} \mathcal{T}_N(d_g+1, d_g)$
6: $h = f^{-1} * g \in R_{N,q}$

Output: Private key (f,g), Public key $(1,h)$

For the sake of completeness we provide sketches of the standardized CCA-2 secure variants of the NTRUEncrypt encryption and decryption algorithms. Further details may be found in [16] and [21], and a reference implementation is freely available [18]. We will need several parameter-set dependent support functions: an invertible message formatting function FMT, a blinding polynomial generation function BGF, and a mask generation function MGF, with types:

$$\begin{aligned}\mathsf{FMT} &: \text{Message} \times \text{Random bits} \to \mathcal{T}_N,\\ \mathsf{BGF} &: \text{Message} \times \text{Random bits} \times \text{Public key} \to \mathcal{P}_N(d_1,d_2,d_3),\\ \mathsf{MGF} &: R_{N,q} \to \mathcal{T}_N.\end{aligned}$$

In practice these three functions are built around hash functions and, heuristically, behave like hash functions themselves. A sketch of the encryption algorithm is provided in Algorithm 2, and a sketch of the decryption algorithm in Algorithm 3.

Algorithm 2 NTRUEncrypt Encryption (sketch)

Input: Public key h, message $M \in \{0,1\}^{mLen}$, and a parameter set with $p = 3$.

repeat
  $b \leftarrow_{\$} \{0,1\}^{bLen}$
  $m' = \mathsf{FMT}(M, b)$
  $r = \mathsf{BGF}(m', b, h)$
  $r' = p \cdot r * h$
  $mask = \mathsf{MGF}(r')$
  $m = m' + mask \pmod p$
until The number of $+1$s, -1s and 0s in m are each $\geq d_m$.
$c = r' + m$

Output: Ciphertext c

Algorithm 3 NTRUEncrypt Decryption (sketch)

Input: Key pair $((f, g), (1, h))$, ciphertext $c \in R_{N,q}$, and a parameter set with $p = 3$.

$a = f * c$
$m = a \pmod p$
$r' = c - m$
$mask' = \mathsf{MGF}(r')$
$m' = m - mask \pmod p$
$(M, b) = \mathsf{FMT}^{-1}(m')$
$r = \mathsf{BGF}(m', b, h)$
if $p \cdot r * h = r'$ and the number of $+1$s, -1s and 0s in m are each $\geq d_m$ **then**
  $result = M$
else
  $result = \perp$
end if

Output: *result*

2 General considerations

Ring parameters: The only restrictions on p and q are that they generate coprime ideals of $\mathbb{Z}[X]/(X^N - 1)$. In this document we will fix $p = 3$ and only consider q that are a power of 2. This choice is motivated by the need for fast arithmetic modulo q, and by the impact of p on decryption failure probability (see Section 6).

For NTRUEncrypt we take N to be prime. Many ideal lattice cryptosystems take N to be a power of two and work in the ring $\mathbb{Z}_q[X]/(X^N + 1)$ primarily because $X^N + 1$ is irreducible over the rationals. Some complications arise from using a reducible ring modulus, but these are easily remedied.

For prime N the ring modulus factors into irreducibles as

$$X^N - 1 = (X - 1)\Phi_N(X)$$

where $\Phi_N(X)$ is the N^{th} cyclotomic polynomial. To maximize the probability that a random f is invertible in $R_{N,q}$ we should choose N such that (2) is inert in the N^{th} cyclotomic field, i.e. we should ensure that $\Phi_N(X)$ is irreducible modulo 2. Such a choice of N ensures that f is invertible so long as $f(1) \neq 0$. It is not

strictly necessary that $\Phi_N(X)$ be irreducibe modulo 2, and one may allow a small number of high degree factors while maintaining a negligible probability of failure per loop iteration in Algorithm 1. A table of reasonable primes is provided in Appendix D.1. Similar considerations apply for other choices of q.

Private key and message parameters: The analysis below will be considerably simpler if we fix the structure of the private key in advance, that is specify how the values d_1, d_2, d_3 and d_g will be derived given N and q. The desire to use a product form value for f comes purely from efficiency considerations, and the use of trinary polynomials comes from a combination of combinatorial considerations and a desire to keep q as small as possible while ensuring that decryption succeeds with all but negligible probability. In order to maximize the size of the key space, while keeping a prescribed number of ± 1s in g, we take $d_g = \lfloor N/3 \rceil$. The expected number of non-zero coefficients in f is $4d_1d_2 + 2d_3$. So, in order to roughly balance the hybrid combinatorial search problems for f and g (Section 4), we take $d_1 \approx d_2 \approx d_3$ with $d_1 = \lfloor \alpha \rceil$ where α is the positive root of $2x^2 + x - N/3$. This gives us $2d_1d_2 + d_3 \approx N/3$.

A Hamming weight restriction is placed on message representatives to avoid significant variation in the difficulty of message recovery due to the choice of representative. Message representatives are trinary polynomials, and we require that the number of $+1$s, -1s and 0s each be greater than d_m. The procedure for choosing d_m is given in Section 5.

3 Review of the hybrid attack

Suppose one is given an NTRU public key $(1, h)$ along with the relevant parameter set. This information determines a basis for a lattice $\mathcal{L}$ of rank $2N$ generated by the rows of

$$L = \left(\begin{array}{c|c} qI_N & 0 \\ \hline H & I_N \end{array}\right) \tag{3}$$

wherein the block H is the circulant matrix corresponding to h, i.e. its rows are the coefficient vectors of $x^i * h$ for $i \in [0, N-1]$. The map $(1, h)R_{N,q} \to \mathcal{L}/q\mathcal{L}$ that sends $(a, b) \mapsto (b_0, \ldots, b_{N-1}, a_0, \ldots, a_{N-1})$ is an additive group isomorphism that preserves the norm defined in Equation 2. As such, if one can find short vectors of $\mathcal{L}$ one can find short elements of the corresponding NTRU module.

The determinant of L is $\Delta = q^N$, giving us a Gaussian expected shortest vector of length $\lambda \approx \sqrt{qN/\pi e}$, though the actual shortest vector will be somewhat smaller than this. A pure lattice reduction attack would attempt to solve Hermite-SVP[4] with factor $\lambda/\Delta^{1/2N} = \sqrt{N/\pi e}$ which is already impractical for N around 100. The experiments of [19] support this claim, they were able to find short vectors in three NTRU lattices with $N = 107$ and $q = 64$ that were generated using binary private keys. Only one of these was broken with BKZ alone, the other two required a heuristic combination of BKZ on the full lattice and BKZ on a projected lattice of smaller dimension with block sizes between 35 and 41.

Consequently the best attacks against NTRUEncrypt tend to utilize a combination of lattice reduction and combinatorial search. In this section we will review one such method from [11], known as the hybrid attack.

The rough idea is as follows. One first chooses $N_1 < N$ and extracts a block, L_1, of $2N_1 \times 2N_1$ coefficients from the center of the matrix L defined in Eq. 3. The rows of L_1 are taken to generate a lattice $\mathcal{L}_1$.

$$\left(\begin{array}{c|c} qI_N & 0 \\ \hline H & I_N \end{array}\right) = \left(\begin{array}{c|c|c} qI_{r_1} & 0 & 0 \\ \hline * & L_1 & 0 \\ \hline * & * & I_{r_2} \end{array}\right) \tag{4}$$

A lattice reduction algorithm is applied to find a unimodular transformation, U', such that $U'L_1$ is reduced, and an orthogonal transformation, Y', is computed such that $U'L_1Y' = T'$ is in lower triangular form. These

[4] In practice q has a strong impact on the effectiveness of pure lattice reduction attacks as well. For large q the relevant problem becomes Unique-SVP which appears to be somewhat easier than Hermite-SVP. Conservative parameter generation should ensure that it is difficult to solve Hermite-SVP to within a factor of $q/\Delta^{1/2N} = \sqrt{q}$.

transformations are applied to the original basis to produce a basis for an isomorphic lattice:

$$T = ULY = \left(\begin{array}{c|c|c} I_{r_1} & 0 & 0 \\ \hline 0 & U' & 0 \\ \hline 0 & 0 & I_{r_2} \end{array}\right) \left(\begin{array}{c|c|c} qI_{r_1} & 0 & 0 \\ \hline * & L_1 & 0 \\ \hline * & * & I_{r_2} \end{array}\right) \left(\begin{array}{c|c|c} I_{r_1} & 0 & 0 \\ \hline 0 & Y' & 0 \\ \hline 0 & 0 & I_{r_2} \end{array}\right) = \left(\begin{array}{c|c|c} qI_{r_1} & 0 & 0 \\ \hline * & T' & 0 \\ \hline * & * & I_{r_2} \end{array}\right). \tag{5}$$

Notice that $(g, f)Y$ is a short vector in the resulting lattice.

In general it is not necessary for the extracted block to be the central $2N_1 \times 2N_1$ matrix, and it is sometimes useful to consider blocks shifted s indices to the top left along the main diagonal. Let $r_1 = N - N_1 - s$ be the index of the first column of the extracted block and $r_2 = N + N_1 - s$ be the index of the final column. The entries on the diagonal of T will have values $\{q^{\alpha_1}, q^{\alpha_2}, \ldots, q^{\alpha_{2N}}\}$, where $\alpha_1 + \cdots + \alpha_{2N} = N$, and the α_i, for i in the range $[r_1, r_2]$, will come very close to decreasing linearly. That is to say, L_1 will roughly obey the geometric series assumption (GSA). The rate at which the α_i decrease can be predicted very well based on the root Hermite factor achieved by the lattice reduction algorithm used.[5] Clearly $\alpha_i = 1$ for $i < r_1$ and $\alpha_i = 0$ for $i > r_2$. By the analysis in Appendix E we expect

$$\alpha_{r_1} = \frac{1}{2} + \frac{s}{2N_1} + 2N_1 \log_q(\delta) \tag{6}$$

$$\alpha_{r_2} = \frac{1}{2} + \frac{s}{2N_1} - 2N_1 \log_q(\delta), \tag{7}$$

and a linear decrease in-between. The profile of the basis will look like one of the examples in Figure 1.

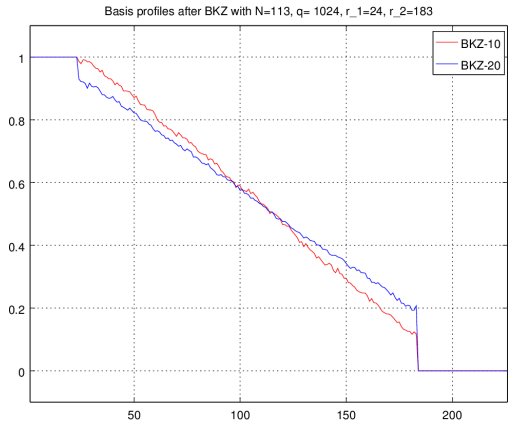

Fig. 1. Log length of i^{th} Gram-Schmidt vector, $\log_q(\|b_i^*\|)$.

By a lemma of Furst and Kannan (Lemma 1 in [11]), if $y = uT + x$ for vectors u and x in $\mathbb{Z}^{2N}$, and $-T_{i,i}/2 < x_i \leq T_{i,i}/2$, then reducing y against T with Babai's nearest plane algorithm will yield x exactly. Thus if v is a shortest vector in $\mathcal{L}$ and $\alpha_{r_2} > \log_q(2\|v\|_\infty)$, it is guaranteed that v can be found by enumerating candidates for its final $K = 2N - r_2$ coefficients. Further knowledge about v can also diminish the search space. For example, if it is known that there is a trinary vector in $\mathcal{L}$, and $\alpha_{r_2} > \log_q(2)$, then applying Babai's nearest plane algorithm to some vector in the set

$$\{(0|v')T - (0|v') : v' \in \mathcal{T}_K\}$$

will reveal it.

[5] A lattice reduction algorithm that achieves root Hermite factor δ returns a basis with $\|\boldsymbol{b}_1\|_2 \approx \delta^n \det(\Lambda)^{1/n}$.

The optimal approach for the attacker is determined by the balancing the cost of combinatorial search on K coordinates against the cost of lattice reduction that results in a sufficiently large α_{2N-K}. Unsurprisingly, naïve enumeration of the possible v' is not optimal.

4 Meet in the middle search

The adaptation of meet-in-the-middle search algorithms to the structure of binary NTRU keys is due to Odlyzko and described in [6]. Generalizations to other private key types are described by Howgrave-Graham in [11]; this is the presentation we follow here. The key idea is to decompose the search space S as $S \subseteq S' \oplus S'$ for some set S' such that $|S'| \approx \sqrt{|S|}$. If s_1 and s_2 are elements of S' such that $s_1 + s_2 = f$, and (f, g) is an element with small coefficients in the NTRU module generated by $(1, h)$, then $(s_1, s_1 * h) = (f, g) - (s_2, s_2 * h)$. In particular, when the coefficients of g are trinary, this implies that $s_1 * h \approx -s_2 * h$ coordinate-wise.

Under the assumption that all approximate collisions can be detected, a meet in the middle search on the full product form NTRUEncrypt key space would require both time and memory of order $O(\sqrt{|\mathcal{P}_N(d_1, d_2, d_3)|})$.

A meet in the middle search is also possible on a basis that has been preprocessed for the hybrid attack as in Equation 5. The assumption that all approximate collisions can be detected will turn out to be untenable in this case, however, in the interest of deriving conservative parameters we will assume that this complication does not arise. Let $\Pi : \mathbb{Z}^N \to \mathbb{Z}^K$ be a projection[6] onto K coordinates of $\mathbb{Z}^N$. Let $P_\Pi = \{v\Pi : v \in \mathcal{P}_N(d_1, d_2, d_3)\}$. The f component of the private key is guaranteed to appear in P_Π, so the expected time and memory required for the attack is $O(\sqrt{|P_\Pi|})$. That said, estimating the size of P_Π is non-trivial.

We may also consider an adversary who attempts this attack on the lattice corresponding to $(1, h^{-1})$ and searches for the g component of the private key instead. This may in fact be the best strategy for the adversary, because while $|\mathcal{P}_N(d_1, d_2, d_3)| < |\mathcal{T}_N(d_g + 1, d_g)|$ for parameters of interest to us, the presence of coefficients not in $\{-1, 0, 1\}$ in product form polynomials leads to a large increase in the relative size of the projected set.

In either case we assume that it is sufficient for the adversary to search for trinary vectors, and that they may limit their search to a projection of $\mathcal{T}_N(d, e)$ for some (d, e). When targeting g we have $d = d_g + 1$, $e = d_g$, and when targeting f we have that both d and e are approximately $2d_1d_2 + d_3$. Clearly when $d = e = N/3$, and $N \gg K$, we should expect that the projection of a uniform random element of $\mathcal{T}_N(d, e)$ onto K coordinates will look like a uniform random element of $\mathcal{T}_K$. For such parameters, the size of the set that must be enumerated in the meet-in-the-middle stage is $\approx 3^{K/2}$.

For $d \neq N/3$, or for large K, not all trinary sequences are equally likely, and the adversary may choose to target a small set of high probability sequences. Consequently we must estimate the size of the set of elements that are typical under the projection. Fix N, K, Π, d, and e and let $\mathcal{S} = \mathcal{T}_N(d, e)$. Let $p : \mathcal{T}_K \to \mathbb{R}$ be the probability mass function on $\mathcal{T}_K$ induced by sampling an element uniformly at random from $\mathcal{S}$ and projecting its coefficient vector onto the set of K coordinates fixed by Π. We will estimate the size of the search space in the hybrid attack as, roughly, $2^{H(p)}$, where $H(p)$ is the Shannon entropy of p.

Let $\mathcal{S}_\Pi(a, b)$ be the subset of $\mathcal{S}$ consisting of vectors, v, such that $v\Pi$ has exactly a coefficients equal to $+1$ and b coefficients equal to -1. By the symmetry of $\mathcal{S}$ under coordinate permutations we have that $p(v\Pi) = p(v'\Pi)$ for all pairs $v, v' \in \mathcal{S}_\Pi(a, b)$. We choose a fixed representative of each type: $v_{a,b} = v\Pi$ for some $v \in \mathcal{S}_\Pi(a, b)$, and write

$$p(v_{a,b}) = \frac{1}{\binom{K}{a}\binom{K-a}{b}} \frac{|\mathcal{S}_\Pi(a, b)|}{|\mathcal{S}|} = \frac{\binom{N-K}{d-a}\binom{N-K-d+a}{d-b}}{\binom{N}{d}\binom{N-d}{d}}. \tag{8}$$

[6] We will abuse notation slightly and allow Π to act on elements of R by acting on their coefficient vectors lifted to $\mathbb{Z}^N$.

As there are exactly $\binom{K}{a}\binom{K-a}{b}$ distinct choices for $v_{a,b}$ this gives us:

$$H(p) = -\sum_{v \in \mathcal{T}_K} p(v) \log_2 p(v) = -\sum_{0 \le a,b \le d} \binom{K}{a}\binom{K-a}{b} p(v_{a,b}) \log_2 p(v_{a,b}). \tag{9}$$

The size of the search space is further decreased by a factor of N since $x^i * g$ is likely to be a distinct target for each $i \in [0, N-1]$. Hence in order to resist the hybrid meet-in-the-middle attack we should ensure

$$\frac{1}{2}(H(p) - \log_2(N)) \ge \lambda. \tag{10}$$

The only variable not fixed by the parameter set itself in Equation 9 is K. In order to fix K we must consider the cost of lattice reduction.

The block to be reduced is of size $(r_2 - r_1) \times (r_2 - r_1)$ where $r_2 = 2N - K$ and $r_1 = \lambda$. Recall that $s = N - (r_1 + r_2)/2$, and $N_1 = (r_2 - r_1)/2$. Having fixed these parameters we can use Equation 7 to determine the strength of the lattice reduction needed to ensure that α_{r_2} is sufficiently large to permit recovery of a trinary vector. In particular, we need

$$\alpha_{r_2} = \frac{N_1 + s}{2N_1} - 2N_1 \log_q(\delta) \ge \log_q(2),$$

which implies that

$$\log_2(\delta) \le \frac{N_1 + s}{4N_1^2} \log_2(q) - \frac{1}{2N_1}. \tag{11}$$

Translating the required root Hermite factor, δ, into a concrete bit-security estimate is notoriously difficult. However there seems to be widespread consensus on the values that are currently out of reach for common security parameters. As such one might use the following step function as a first approximation:

$$\delta^*(\lambda) = \begin{cases} 1.009 & \text{if } \lambda \le 60 \\ 1.008 & \text{if } 60 < \lambda \le 80 \\ 1.007 & \text{if } 80 < \lambda \le 128 \\ 1.005 & \text{if } 128 < \lambda \le 256 \\ 1 & \text{otherwise.} \end{cases}$$

A more refined approach involving the simulator from [1] is used in Section 8.

Rewriting Equation 11 in terms of N, q, r_1 and K we define:

$$\log_2(\eta(N, q, r_1, K)) = \frac{(N - r_1)\log_2(q)}{4N^2 - 4N(K + r_1) + (K^2 + 2r_1K + r_1^2)} - \frac{1}{2N - (K + r_1)}. \tag{12}$$

A parameter set resists hybrid meet-in-the-middle attacks on private keys if Equation 10 is satisfied and

$$1 < \eta(N, q, r_1, K) \le \delta^*(r_1). \tag{13}$$

5 Rejecting sparse (and dense) message representatives

The parameter sets in this paper specify the exact number of 1's and -1's in each of r_1, r_2, r_3, which reveals the quantity $r(1)$, that is, the polynomial r evaluated at 1. As an encrypted message has the form $e = pr * h + m$, the value $m(1)$ modulo q is revealed by the known quantities $r(1), e(1), h(1)$. The value $m(1)$ in turn reveals the difference between the number of 1's and the number of -1's in the message representative.

We assume that the randomness used in querying MGF is sufficent to ensure that the masking polynomial generated in Line 6 of Algorithm 2 is sampled uniformly from $\mathcal{T}_N$. This results in a uniformly distributed

message representative. The expected value of $m(1)$ is zero, but for large $|m(1)|$, the size of the search space for m decreases, making a meet in the middle search for (r, m) easier. We assume that the adversary observes a very large number of messages and can freely condition their attack on the value of $m(1)$ regardless of the probability that a uniform random message representative takes that value.

Line 8 of Algorithm 2 and Line 8 of Algorithm 3 work to prevent these types of attacks by forbidding message representatives with particularly small, or large, Hamming weight. Specifically, these lines forbid the number of $+1$s, -1s, or 0s to be less than a given parameter, d_m. The choice of d_m depends primarily affects resistance against hybrid combinatorial attacks, but d_m also has an impact on decryption failure probability, as will be discussed in Section 6.

The calculation for determining resistance against hybrid combinatorial attacks is very similar to that leading up to Equation 9, but there are two key differences. First, in Section 4 we were primarily concerned with validating the security of the obvious choice $d_g = \lfloor N/3 \rceil$. Here we will need to search for d_m. Second, having fixed d_m we need to condition the distribution of projected elements on the value of $m(1)$.

The search space for d_m can be constrained by imposing an arbitrary upper bound on the probability of a failure in Line 8 of Algorithm 2. Such a failure is roughly as expensive as a full encryption, so d_m should be chosen to ensure that failures are rare.

Let

$$I(d_m) = \{(i, j) : d_m \leq i < (N - 2d_m),\ d_m \leq j < (N - d_m - i)\}.$$

We will only consider d_m satisfying:

$$2^{-10} \geq 1 - 3^{-N} \left(\sum_{(i,j) \in I(d_m)} \binom{N}{i} \binom{N-i}{j} \right) \tag{14}$$

Let K be the value derived in Section 4. Fix Π and let $\mathcal{S}(e_1, e_2; a, b)$ be the set of projections of elements of $\mathcal{T}(e_1, e_2)$ with a ones and b minus ones. Let $\mathcal{M}$ be the subset of $\mathcal{T}_N$ satisfying the d_m constraint. Let $p : \mathcal{T}_K \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{R}$ be the probability mass function given by

$$p(v, e_1, e_2) = \text{Prob}_{m \leftarrow_{\$} \mathcal{M}} \left(m\Pi = v \text{ and } m \in \mathcal{T}_N(e_1, e_2) \right),$$

i.e. $p(v, e_1, e_2)$ is the probability that an m sampled uniformly from $\mathcal{M}$ has e_1 ones, e_2 minus ones, and is equal to v under projection. If the information leakage from $m(1)$ determined e_1 and e_2 then we could use essentially the same analysis as Section 4 and our security estimate would be

$$\frac{1}{2} \min_{(e_1, e_2) \in I(d_m)} H(p_{d_m} | e_1, e_2).$$

However the adversary only learns $m(1) = e_1 - e_2$, so we will account for their uncertainty about whether $m \in \mathcal{T}(e_1, e_2)$ given $m(1) = e_1 - e_2$.

The marginal distribution on e_1 and e_2 conditioned on the event $m(1) = y$ is

$$\begin{aligned} q(e_1, e_2 | m(1) = y) &= \sum_{v \in \mathcal{T}_K} p(v, e_1, e_2 | m(1) = y) \\ &= \binom{N}{e_1} \binom{N - e_1}{e_2} \left(\sum_{\substack{i - j = y \\ (i,j) \in I(d_m)}} \binom{N}{i} \binom{N-i}{j} \right)^{-1}. \end{aligned}$$

As such we will consider a parameter set secure against hybrid meet-in-the-middle attacks on messages provided that:

$$\lambda \ \leq \ \min_{y} \min_{\substack{e_1 - e_2 = y \\ (e_1, e_2) \in I(d_m)}} \frac{1}{2} H(p | e_1, e_2) - \log_2 q(e_1, e_2 | m(1) = y). \tag{15}$$

Evaluating this expression is considerably simplified by noting that local minima will be found at the extremal points: $|e_1 - e_2| = N - 3d_m$ and $e_1 = e_2 \approx N/3$.

Note that unlike the estimate in Section 4 we do not include a $-\log_2(N)$ term to account for rotations of m.

6 Estimating the probability of decryption failure

As remarked earlier, in order for decryption to succeed the coefficients of

$$a = p * (r * g + m * F) + m \tag{16}$$

must have absolute value less than $q/2$.

Assuming $p \in \mathbb{Z}$, and trinary g and m, the triangle inequality yields:

$$\|a\|_\infty \leq p\left(\|r\|_1\|g\|_\infty + \|F\|_1\|m\|_\infty\right) + 1 = p\left(\|r\|_1 + \|F\|_1\right) + 1. \tag{17}$$

Thus with product form r and F decryption failures can be avoided entirely by ensuring $(q-2)/2p > 8d_1d_2 + 4d_3$. However, since ciphertext expansion scales roughly as $N\log_2(q)$, it can be advantageous to consider probabilistic bounds as well. The probability

$$\text{Prob}\,(\text{ a given coefficient of } r * g + m * F \text{ has absolute value } \geq c) \tag{18}$$

can be analyzed rather well by an application of the central limit theorem. This was done for the case of trinary r, g, m, F in [3]. Here we provide a modified analysis for the case where the polynomials r and F take a product form. In particular, we assume that $r = r_1 * r_2 + r_3$, $F = F_1 * F_2 + F_3$, where each r_i and F_i has exactly d_i coefficients equal to 1, d_i coefficients equal to -1, and the remainder equal to 0.

Let X_k denote a coefficient of $r * g + m * F$. The spaces from which r and m are drawn are invariant under permutations of indices, so the probability that $|X_k| > c$ does not depend on the choice of k.[7] Note that X_k has the form

$$X_k = (r_1 * r_2 * g)_k + (r_3 * g)_k + (F_1 * F_2 * m)_k + (F_3 * m)_k, \tag{19}$$

and each term in the sum is itself a sum of either $4d_1d_2$ or $2d_3$ (not necessarily distinct) coefficients of g or m. For instance,

$$(r_1 * r_2 * g)_k = \sum_{i,j} (r_1)_i (r_2)_j (g)_{(k-i-j)}$$

and only the $4d_1d_2$ pairs of indices corresponding to non-zero coefficients of r_1 and r_2 contribute to the sum. We can think of each index pair as selecting a sign $\epsilon(i)$ and an index $a(i)$ and rewrite the sum as

$$(r_1 * r_2 * g)_k = \sum_{i=1}^{4d_1d_2} \epsilon(i)(g)_{a(i)}.$$

While the terms in this sum are not formally independent (since a may have repeated indices, and g has a prescribed number of non-zero coefficients) extensive experiments show that the variance of $(r_1 * r_2 * g)_k$ is still well approximated by treating $(g)_{a(i)}$ as a random coefficient of g, i.e. as taking a non-zero value with probability $(2d_g + 1)/N$:

$$\mathbb{E}\left[(r_1 * r_2 * g)_k^2\right] \approx \sum_{i=1}^{4d_1d_2} \mathbb{E}\left[(\epsilon(i)(g)_{a(i)})^2\right] = \sum_{i=1}^{4d_1d_2} \mathbb{E}\left[(g)_{a(i)}^2\right] = 4d_1d_2 \cdot \frac{2d_g + 1}{N}$$

[7] The X_k for different k have the same distribution, but they are not completely independent. However, they are so weakly correlated as to not affect our analysis.

Nearly identical arguments can be applied to compute the variances of the other terms of Eq. 19, although some care must be taken with the terms involving m. While an honest party will choose m uniformly from the set of trinary polynomials, m could be chosen adversarily to maximize its Hamming weight and hence the probability of a decryption failure. Due to the d_m constraint (Section 5), the number of non-zero coefficients of m cannot exceed $N - d_m$. As such we model the coefficients of m as taking ± 1 each with probability $(1 - d_m/N)$ and 0 with probability d_m/N.

With these considerations the variance of $(r_1 * r_2 * g)_k + (r_3 * g)_k$ is found to be $\sigma_1^2 = (4d_1d_2 + 2d_3) \cdot \frac{2d_g+1}{N}$, and the variance of $(F_1 * F_2 * m)_k + (F_3 * m)_k$ is found to be $\sigma_2^2 = (4d_1d_2 + 2d_3) \cdot (1 - \frac{d_m}{N})$. Both terms are modeled as sums of i.i.d. random variables, and the d_i are chosen such that $4d_1d_2 + 2d_3 \approx 2N/3$, so for sufficiently large N the central limit theorem suggests that each term will have a normal distribution. Finally X_k can be expected to be distributed according to the convolution of these two normal distributions, which itself is a normal distribution with variance

$$\sigma^2 = \sigma_1^2 + \sigma_2^2 = (4d_1d_2 + 2d_3) \cdot \frac{N - d_m + 2d_g + 1}{N}. \tag{20}$$

The probability that a normally distributed random variable with mean 0 and standard deviation σ exceeds c in absolute value is given by the complementary error function, specifically $\texttt{erfc}(c/(\sqrt{2}\sigma))$. Applying a union bound, the probability that any of the N coefficients of $r * g + m * f$ is greater than c is bounded above by $N \cdot \texttt{erfc}(c/(\sqrt{2}\sigma))$.

With respect to the security parameter, λ, this imposes the constraint

$$N \cdot \texttt{erfc}((q-2)/(2\sqrt{2} \cdot p \cdot \sigma)) < 2^{-\lambda} \tag{21}$$

where $\sigma = \sigma(N, d_1, d_2, d_3, d_g, d_m)$ as in Eq. 20.

7 Product form combinatorial strength

The search space for a triple of polynomials F_1, F_2, F_3 where each polynomial F_i has d_i 1's and d_i -1's is of size:

$$|\mathcal{P}_N(d_1, d_2, d_3)| = \binom{N}{d_1}\binom{N-d_1}{d_1}\binom{N}{d_2}\binom{N-d_2}{d_2}\binom{N}{d_3}\binom{N-d_3}{d_3}.$$

Thus a purely combinatorial meet-in-the-middle search on product form keys can be performed in time and space

$$O(\sqrt{|\mathcal{P}_N(d_1, d_2, d_3)| / N}).$$

Where we have divided by N to account for the fact that rotations of a given triple are equivalent.

Finally, one could construct a $3N$ dimensional lattice attack by considering the lattice generated by linear combinations of the vectors $(1, 0, f_1 * h), (0, 1, h), (0, 0, q)$, where each entry corresponds to N entries in the lattice. The vector (f_2, f_3, g) will be a very short vector, but the increase of the dimension of the lattice by N, without any corresponding increase in the determinant of the lattice, leads to a considerably harder lattice reduction problem. As this attack also requires a correct guess of f_1 we will not consider it further.

8 Sample parameter generation

Appendix A gives an explicit algorithm for deriving parameters. We will ignore the implicit outer loop over security parameters and consider the case of $N = 401$ starting from Line 3.

Our recommendations for the key structure suggests taking $d_g = 134$, $d_1 = 8$, $d_2 = 8$, $d_3 = 6$. Taking $d_m = 102$ satisfies Equation 14 with a probability of $2^{-10.4}$ of rejecting a message representative due to its coefficient sum. A direct meet-in-the-middle attack on the product form key space will involve testing approximately 2^{145} candidates. As this is an upper bound on the security of the parameter set we will ensure

that our decryption failure probability is less than 2^{-145}. This implores us to take $q = 2048$, for which there is, by Equation 21, a decryption failure probability of 2^{-217}.

In order to finish the parameter derivation we need a tighter estimate on its security. It may be significantly less than 145, in which case we may be able to reduce q.

We estimate the security of the parameter set by minimizing the adversary's expected cost over choices of the hybrid attack parameter K. Equation 12 specifies, for each K, the root Hermite factor, δ, that must be reached during the lattice reduction phase of the hybrid attack in order for the combinatorial stage to be successful. We use the BKZ-2.0 simulator of [1] to determine the blocksize and number of rounds of BKZ that will be required to reach root Hermite factor δ.

To turn the blocksize and iteration count into a concrete security estimate we need estimates on the number of nodes visited per call to the enumeration subroutine of BKZ. Table 8 summarizes upper bounds given by Chen and Nguyen in [1] and in the full version of the same paper[2]. The estimates of the full version are significantly lower than the original, and have perhaps not recieved the same scrutiny. In what follows we will consider the implications of both estimates.

β	100	110	120	130	140	150	160	170	180	190	200	210	220	230	240	250
Estimate 1[1]	41.4	47.1	53.1	59.8	66.8	75.2	84.7	94.7	105.8	117.6	129.4	-	-	-	-	204.1
Estimate 2[2]	39	44	49	54	60	66	72	78	84	96	99	105	111	120	127	134

Table 2. Upper bounds on $\log_2$ number of nodes enumerated in one call to enumeration subroutine of BKZ-2.0 as reported in the original and full versions of the paper.

To facilitate computer search for parameters we fit curves to the estimates in Table 8, and following [1] we estimate the per-node cost, as 2^7 operations. The resulting predictions for the cost of the lattice reduction stage, in terms of the blocksize, the dimension of the sublattice to be reduced, and number of rounds are thus:

$$\begin{aligned} \text{LogNodes}_1(\beta) &= 0.00405892\beta^2 - 0.337913\beta + 34.9018 \\ \text{Estimate}_1(\beta, dim, \#rounds) &= \text{LogNodes}_1(\beta) + \log_2(dimension \cdot \#rounds) + 7 \end{aligned}$$

$$\begin{aligned} \text{LogNodes}_2(\beta) &= 0.000784314\beta^2 + 0.366078\beta - 6.125 \\ \text{Estimate}_2(\beta, dim, \#rounds) &= \text{LogNodes}_2(\beta) + \log_2(dimension \cdot \#rounds) + 7. \end{aligned}$$

Finally our security estimate requires a search over K to balance the cost of lattice reduction against the cost of combinatorial search given by Equation 10.

We find that for $K = 166$ the BKZ-2.0 simulator suggests that 11 rounds of BKZ-181 will achieve the requisite root hermite factor of $\delta = 1.0067$. Using Estimate_1 this reduction will require 2^{126} operations, matching the cost of 2^{127} given by Equation 10.

Similarly, for $K = 154$ the BKZ-2.0 simulator suggests that 10 rounds of BKZ-197 will achieve to the requisite $\delta = 1.0064$. Using Estimate_2 this reduction will require 117 operations, matching the cost of 2^{116} given by Equation 10.

Using either of the BKZ estimates we find that we cannot decrease q without violating the constraint on the decryption failure probability, and we are done.

The parameter set we have just (re-)derived originally appeared in the EESS #1 standard at the 112 bit security level. All four product-form parameter sets from EESS #1 are reviewed in Table 3 with security estimates following the above analysis. Note that while the algorithm in Appendix A rederives the $N = 401$ parameter set almost exactly (d_g is 133 in EESS #1), this is not true for the $N = 593$ and $N = 743$ parameter sets. In particular, all four of the published parameter sets take $q = 2048$, and this does not lead to a formally negligible probability of decryption failure for $N = 593$ or $N = 743$. Note also that the number of prime ideals lying above (2) is more than recommended for $N = 439$ and $N = 593$. Table 3 presents

security estimates for the standardized parameters rather than those that would be output by the algorithm of Appendix A.

EESS #1 Parameter Sets and Security Estimates													
Original								Estimate 1		Estimate 2		Product form	$\log_2$
security est.	N	q	d_1	d_2	d_3	d_g	d_m	K	Cost	K	Cost	search cost	dec. fail prob.
112	401	2048	8	8	6	133	101	166	127	154	**116**	145	-217
128	439	2048	9	8	5	146	112	192	145	175	**133**	147	-195
192	593	2048	10	10	8	197	158	303	236	264	204	**193**	-139
256	743	2048	11	11	15	247	204	423	330	360	280	**256**	-112

Table 3.

References

1. Yuanmi Chen and Phong Q Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT 2011*, pages 1–20. Springer, 2011.
2. Yuanmi Chen and Phong Q Nguyen. BKZ 2.0: Better lattice security estimates (Full Version). Available at http://www.di.ens.fr/ ychen/research/Full_BKZ.pdf
3. P. Hirschhorn, J. Hoffstein, N. Howgrave-Graham, W. Whyte, Choosing NTRU parameters in light of combined lattice reduction and MITM approaches, Applied Cryptography and Network Security, LNCS, Volume 5536/2009, 437–455
4. 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
5. J. Hoffstein, J.H. Silverman, Optimizations for NTRU, Public Key Cryptography and Computational Number Theory (Warsaw, Sept. 11–15, 2000), Walter de Gruyter, Berlin–New York, 2001, 77–88.
6. J. Hoffstein, J.H. Silverman, W. Whyte, Meet-in-the-middle Attack on an NTRU private key, Technical report, NTRU Cryptosystems, July 2006. Report #04, available at `http://www.ntru.com`.
7. J. Hoffstein, J.H. Silverman, Provable Probability Bounds for NTRUEncrypt Convolution Wrap/Gap Events Technical report, NTRU Cryptosystems, July 2007. Report #022, available at `http://www.ntru.com`.
8. J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. Silverman, W. Whyte, *NTRUSign: Digital Signatures Using the NTRU Lattice*, CT-RSA 2003.
9. J. Hoffstein, J. Silverman, *Random small hamming weight products with applications to cryptography*, Special issue on the 2000 com2Mac workshop, 2003, 37-49.
10. Jeff Hoffstein, Nicholas Howgrave-graham, Jill Pipher, Joseph H. Silverman, William Whyte, Performance Improvements and a Baseline Parameter Generation Algorithm for NTRUSign, In Proc. of Workshop on Mathematical Problems and Techniques in Cryptology, 2005, pp 9 9–126
11. N. Howgrave-Graham, *A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU*, Lecture Notes in Computer Science, pringer Berlin / Heidelberg, in Advances in Cryptology - CRYPTO 2007, Volume 4622/2007, pages 150–169.
12. N. Howgrave-Graham, J. H. Silverman, W. Whyte Choosing Parameter Sets for NTRUEncrypt with NAEP and SVES-3, Topics in cryptology—CT-RSA 2005, 118–135, Lecture Notes in Comput. Sci., 3376, Springer, Berlin, 2005. `http://www.ntru.com/cryptolab/articles.htm\#2005\_1`
13. P. Hirschhorn, J. Hoffstein, N. Howgrave-Graham, W. Whyte, *C*hoosing NTRUEncrypt Parameters in Light of Combined Lattice Reduction and MITM Approaches, ACNS 2009. 437-455
14. P. Nguyen, O. Regev, *Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures*, Eurocrypt 2006, 271-288.
15. V. Shoup *NTL: A Library for doing Number Theory*, Version 5.4, `http://www.shoup.net/ntl`
16. IEEE Std 1363.1-2008, IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices, 2008
17. L. Ducas, A. Durmus, T. Lepoint, V. Lyubashevsky, *Lattice Signatures and Bimodal Gaussians*, CRYPTO 2013, 40-56.

18. *NTRUOpenSourceProject*, `https://github.com/NTRUOpenSourceProject/ntru-crypto`
19. N. Gama, P. Nguyen, *Predicting lattice reduction*, EUROCRYPT'08, 31-51
20. D. Stehl, R. Steinfeld, *Making NTRU as Secure as Worst-Case Problems over Ideal Lattices*, EUROCRYPT 2011, 27-47
21. Consortium for Efficient Embedded Security, *Efficient Embedded Security Standard (EESS) #1 version 3.0*, 2015, available at `https://github.com/NTRUOpenSourceProject/ntru-crypto`.
22. *NTRU challenge*, Feb 2015, `https://www.ntru.com/ntru-challenge/`.

A Explicit algorithm for computing parameters

The following algorithm determines the smallest recommended N from Table 4 or Table 5 that allows for k bit security. Additional details, such as recommendations on how to efficiently perform the search in Line 16, may be found in our implementation which is available at `https://github.com/NTRUOpenSourceProject/ntru-params`.

Algorithm 4 NTRUEncrypt parameter generation

Input: Desired security level k.

1: Let n_j be the j^{th} value, ordered by magnitude, from either Table 4 or Table 5.
2: Set $j = 1$.
3: Set $N = n_j$.
4: Set $d_g = \left\lfloor \frac{N}{3} \right\rceil$.
5: Set $d_1 = \left\lceil \frac{1}{4} \left(\sqrt{1 + \frac{8N}{3}} - 1 \right) \right\rceil$ {The next integer above the positive root of $2x^2 + x - N/3$.}
6: Set $d_2 = \left\lceil \left(\frac{N}{3} - d_1 \right) / (2d_1) \right\rceil$.
7: Set $d_3 = \max\left(\left\lceil \frac{d_1}{2} + 1 \right\rceil, \left\lceil \frac{N}{3} - 2d_1 d_2 \right\rceil \right)$.
8: Set d_m to be the largest value satisfying Equation 14.
9: Set $k_1 = \left\lfloor \frac{1}{2} \log_2 \left(|\mathcal{P}_N(d_1, d_2, d_3)| / N \right) \right\rfloor$. {Cost of direct combinatorial search gives an upper bound on the security.}
10: **if** $k_1 < k$ **then**
11: Increment j.
12: Goto line 3.
13: **end if**
14: Set σ according to Equation 20.

$$\sigma = \left((4d_1 d_2 + 2d_3) \cdot \frac{N - d_m + 2d_g + 1}{N} \right)^{1/2}.$$

15: Set q to be the smallest power of 2 satisfying

$$N \cdot \texttt{erfc}\left((q-2)/(6\sqrt{2}\sigma) \right) < 2^{-k_1}.$$

{Estimate security}
16: Search for a hybrid parameter K that minimizes the maximum of the cost estimates for hybrid attacks. Equation 12 gives the cost of the lattice reduction, and Equations 10 and 15 give the cost of combinatorial search for key- and message-recovery attacks respectively. Let k_2 be the corresponding security estimate.
17: **if** $k > \min(k_1,\ k_2)$ **then**
18: Increment j.
19: Go to Line 3.
20: **end if**
21: Let $q' = q/2$.
22: **if** $N \cdot \texttt{erfc}\left((q'-2)/(6\sqrt{2}\sigma) \right) < 2^{-k}$ **then**
23: Set $q = q'$
24: Repeat security estimate (Line 16) with modulus q' and set k_2 equal to the result.
25: Go to Line 17.
26: **end if**

Output: $[N, q, d_1, d_2, d_3, d_g, d_m]$.

B Security estimates for NTRU challenges

NTRU Challenge Parameter Sets and Security Estimates												
							Estimate 1		Estimate 2		Product form	$\log_2$
N	q	d_1	d_2	d_3	d_g	d_m	K	Cost	K	Cost	search cost	dec. fail prob.
107	512	4	4	4	36	20	-	-	-	-	63	-46
113	1024	5	4	3	38	21	-	-	-	-	63	-162
131	1024	5	4	4	44	26	-	-	-	-	71	-159
139	1024	5	5	3	46	28	-	-	-	-	72	-132
149	1024	5	5	3	50	31	-	-	-	-	73	-132
163	1024	5	5	4	54	35	-	-	50	35	80	-128
173	1024	6	5	4	58	37	61	43	53	37	86	-109
181	1024	6	5	4	60	39	65	46	56	40	87	-109
191	1024	6	5	4	64	42	67	49	61	43	88	-109
199	1024	6	5	6	66	44	70	50	65	46	99	-91
211	1024	6	6	4	70	48	74	53	70	50	95	-92
227	1024	6	6	4	76	52	80	58	77	57	97	-92
239	1024	7	6	4	80	56	85	63	83	61	103	-79
251	1024	7	6	4	84	59	91	67	89	66	104	-79
263	1024	7	6	4	88	62	97	72	95	69	105	-79
271	1024	7	6	6	90	65	102	75	99	73	117	-76
281	2048	7	7	4	94	67	97	71	94	69	112	-284
293	2048	7	7	4	98	71	103	76	100	74	113	-285
307	2048	7	7	4	102	75	110	82	107	79	114	-283
317	2048	8	7	5	106	78	116	86	112	83	126	-249
331	2048	8	7	5	110	82	123	93	125	93	128	-249
347	2048	8	7	5	116	86	133	99	141	105	129	-249
359	2048	8	7	8	120	90	140	106	148	111	147	-243
367	2048	8	8	5	122	92	150	112	136	103	136	-219
379	2048	8	8	5	126	96	153	115	142	107	137	-219
389	2048	8	8	5	130	99	159	119	147	111	138	-219
401	2048	8	8	6	134	102	166	127	154	116	145	-217

C Other known attacks

The hybrid attack is by far the most effective attack against NTRU cryptosystems. Nevertheless, there are other known attacks that need to be considered.

C.1 A combinatorial attack over $\mathbb{F}_2$

Now we describe a combinatorial attack that makes use of special structure of secret polynomials f and g. This attack also relies on q being an even number. For simplicity, we use $d_f = \lfloor \frac{N}{3} \rfloor$ to demonstrate this attack.

Recall that we have

$$f * h = g \bmod (X^N - 1) \bmod q.$$

In our parameter setting, q is always an even number. Therefore, we also have

$$f * h = g \bmod (X^N - 1) \bmod 2.$$

This suggests that instead of conducting a combinatorial search on all possible f (which consists of roughly $\lfloor \frac{N}{3} \rfloor$ of 1s, 0s and -1s), one can search for all possible $f \bmod 2$, that consists of $2\lfloor \frac{N}{3} \rfloor$ of 1s and $(N - 2\lfloor \frac{N}{3} \rfloor)$ of 0s. In particular, since there are more 1s than 0s for these parameter sets, we try to identify the positions of 0s rather than 1s.

To do so, one constructs a list of possible binary polynomials where there are roughly $\frac{N}{6}$ zero coefficients. Birthday paradox tells us that when this list is big enough, one can expect to have two polynomials in this list, namely f_1 and f_2, such that

$$\mathsf{f}_1 + \mathsf{f}_2 = x^i f \bmod 2$$

for some i between 0 and $N - 1$.

The next task is to detect such collusion when it occurs. The special structure of g helps us with this task. When f_1 and f_2 are random ones, we assume that the least significant bits of $(\mathsf{f}_1 + \mathsf{f}_2)h$ follows binomial distribution, while if f_1 and f_2 forms (a cyclic rotation of) secret polynomial f, then $(\mathsf{f}_1 + \mathsf{f}_2)h$ gives (a cyclic rotation of) secret polynomial g. Consequently, one can simply check $(\mathsf{f}_1 + \mathsf{f}_2)h \bmod 2$ for collusion.

The above procedure allows us to distinguish 0 coefficients from non-zeros in f. Once we have located positions of all 0, we also need to separate positive coefficients from negative ones. This can be done by repeat the above procedure with minor modification. We omit the details, although we remark that the cost of this step cannot exceed the cost of the previous step, i.e., locating 0s.

In our parameter setting, we have ensured that our parameter sets are secure against this attack for corresponding security parameter. For example, there are approximately $\binom{N}{\lfloor \frac{N}{6} \rfloor}$ distinct polynomials, among which there are around $N\binom{\lfloor \frac{N}{3} \rfloor}{\lfloor \frac{N}{6} \rfloor}$ good candidates, i.e., polynomials that is a potion of f. This figure comes from the fact that 1) there are N polynomials that reveals secret polynomial due to cyclic rotations, and 2) there are roughly $\binom{\lfloor \frac{N}{3} \rfloor}{\lfloor \frac{N}{6} \rfloor}$ ways to split each polynomial of $\lfloor \frac{N}{3} \rfloor$ zero coefficients into two polynomials with $\lfloor \frac{N}{6} \rfloor$ zero coefficients. There will be duplicates among those good candidates, which we simply neglects to give more advantage to the attacker. As a result, if one construct a list of $\binom{N}{\lfloor \frac{N}{6} \rfloor} / \sqrt{\binom{\lfloor \frac{N}{3} \rfloor}{\lfloor \frac{N}{6} \rfloor}}$ distinct polynomials, he can hope to have a collusion with the probability of roughly 50%.

In a more general setting where d_f is not necessarily $\lfloor \frac{N}{3} \rfloor$, we need to have

$$\frac{\binom{N}{\frac{N}{2} - d_f}}{\sqrt{N\binom{N-2d_f}{\frac{N}{2} - d_f}}} > 2^k$$

when $d_f \geq \frac{N}{4}$ and

$$\frac{\binom{N}{d_f}}{\sqrt{N\binom{2d_f}{d_f}}} > 2^k$$

when $d_f \leq \frac{N}{4}$, where k is the security parameter. The second condition comes from the cases where f is very sparse, and there is less number of ± 1s coefficients combined than 0 coefficient, in which cases we target the positions of the non-zero coefficients rather than 0s.

Another reason that makes this combinatorial attack unsuccessful is the rate of false positives. We detect collusions by checking the distribution of $(\mathsf{f}_1 + \mathsf{f}_2)h \bmod 2$. We have assume that when $(\mathsf{f}_1 + \mathsf{f}_2 \bmod 2)$ is not (a cyclic rotation) equal to $(f \bmod 2)$, then its coefficients follow the Binomial distribution. However, even though most of samples of $\mathsf{f}_1 + \mathsf{f}_2$ will have almost balanced number of 1 and 0 coefficients, there are still a very large number of samples that have the exact same format as g. Those are the false positive results, i.e., a collusion is observed due to probability rather than an actual hit on the secret polynomial. Indeed, when g is in a balanced form (with around $\lfloor \frac{N}{3} \rfloor$ 1s, 0s and -1s), the number of false positive samples is so much greater than correct collusions, the effect to distinguish a sound sample from false positives is beyond the number of operations allowed to an attacker.

Here is some final remarks. This attack works better in the case when either f or g is very sparse. This attack can also be used to locate g rather than f. All of our parameters are secure against this type of attacks.

D Tables

D.1 N suitable for use when q is a power of two

Let N be prime, let q be a power of 2, and let m be the order of 2 in $(\mathbb{Z}/N\mathbb{Z})^*$. The N^{th} cyclotomic polynomial, $\Phi_N(X) = (X^N - 1)/(X - 1)$ has $(N - 1)/m$ degree m irreducible factors mod 2. There are, consequently, $(2^m - 1)^{(N-1)/m}$ invertible elements in $\mathbb{Z}_2[X]/(\Phi(X))$. Provided that m is sufficiently large, say $N - 1$ or $(N - 1)/2$, the probability that a randomly chosen element is non-invertible is negligible.

Furthermore, while no attacks have been proposed that would exploit the prime ideal factorization of (2), it seems prudent to avoid additional algebraic structure whenever possible.

With these considerations in mind we suggest that the N parameter be chosen from one of the following tables when q is taken to be a power of 2.

101,	107,	131,	139,	149,	163,	173,	179,	181,	197,
211,	227,	269,	293,	317,	347,	349,	373,	379,	389,
419,	421,	443,	461,	467,	491,	509,	523,	541,	547,
557,	563,	587,	613,	619,	653,	659,	661,	677,	701,
709,	757,	773,	787,	797,	821,	827,	829,	853,	859,
877,	883,	907,	941,	947,	1019,	1061,	1091,	1109,	1117,
1123,	1171,	1187,	1213,	1229,	1237,	1259,	1277,	1283,	1291,
1301,	1307,	1373,	1381,	1427,	1451,	1453,	1483,	1493,	1499,
1523,	1531,	1549,	1571,	1619,	1621,	1637,	1667,	1669,	1693,
1733,	1741,	1747,	1787,	1861,	1867,	1877,	1901,	1907,	1931.

Table 4. First 100 primes > 100 for which $\mathrm{ord}_{(\mathbb{Z}/N\mathbb{Z})^*}(2) = (N - 1)$, i.e. (2) is inert

103,	137,	167,	191,	193,	199,	239,	263,	271,	311,
313,	359,	367,	383,	401,	409,	449,	463,	479,	487,
503,	521,	569,	599,	607,	647,	719,	743,	751,	761,
769,	809,	823,	839,	857,	863,	887,	929,	967,	977,
983,	991,	1009,	1031,	1039,	1063,	1087,	1129,	1151,	1223,
1231,	1279,	1297,	1303,	1319,	1361,	1367,	1409,	1439,	1447,
1487,	1489,	1511,	1543,	1559,	1567,	1583,	1607,	1663,	1697,
1759,	1783,	1823,	1847,	1871,	1873,	1879,	1951,	1993,	2039.

Table 5. First 80 primes > 100 for which $\mathrm{ord}_{(\mathbb{Z}/N\mathbb{Z})^*}(2) = (N - 1)/2$, i.e. (2) is a product of two prime ideals

E Height of the "cliff"

Given a basis for a lattice of rank m an algorithm that achieves root Hermite factor δ returns a basis with $\|\boldsymbol{b}_1\| \approx \delta^m \det(\Lambda)^{1/m}$.

Under the Geometric Series Assumption (GSA) we have

$$\|\boldsymbol{b}_i^*\| = \eta^{-(i-1)}\|\boldsymbol{b}_1^*\| = \eta^{-(i-1)}\|\boldsymbol{b}_1\| = \eta^{-(i-1)}\delta^m \det(\Lambda)^{1/m}.$$

Furthermore

$$1 = \frac{1}{\det(\Lambda)} \prod_{i=1}^{m} \|\boldsymbol{b}_i^*\| = \eta^{-m(m-1)/2}\ \delta^{m^2}.$$

So,

$$\eta = \delta^{2m/(m-1)}.$$

In the hybrid attack setting we reduce a rank $m = 2N_1$ lattice that has determinant q^{N_1+s}. Let $r_1 = N - N_1 - s$ and $r_2 = N + N_1 - s$ and let $\alpha_i = \log_q(\|b_i^*\|)$. The height of the cliff is given by α_{r_2}. Under the assumption that $\alpha_{r_1} = 1$ the cliff is approximated by $1 - 2N_1 \log_q(\eta) \approx 1 - 4N_1 \log_q(\delta)$. However we can slightly improve this approximation by taking the determinant constraint into consideration. In practice there is no guarantee that $\alpha_{r_1} = 1$, in fact with strong reduction there will be another "cliff" at the beginning of the reduced block. Assuming that the slope, η, is fixed by the basis reduction, we estimate α_{r_1} by requiring that $\sum_{i=r_1}^{r_2} \alpha_i = N_1 + s$. In which case

$$\begin{aligned} N_1 + s &= \sum_{i=r_1}^{r_2} \alpha_i \\ &= \sum_{i=r_1}^{r_2} \left(\alpha_{r_1} - (i - r_1)\log_q(\eta)\right) \\ &= 2N_1\alpha_{r_1} - N_1(2N_1+1)\log_q \eta. \end{aligned}$$

Which implies that

$$\begin{aligned} \alpha_{r_1} &= \frac{N_1 + s}{2N_1} + \frac{N_1(2N_1-1)}{2N_1}\log_q(\eta) \\ &= \frac{N_1 + s}{2N_1} + \frac{N_1(2N_1-1)}{2N_1}\frac{4N_1}{2N_1-1}\log_q(\delta) \\ &= \frac{1}{2} + \frac{s}{2N_1} + 2N_1\log_q(\delta). \end{aligned}$$

In actuality, $\alpha_{r_1} = \min(1,\ \frac{1}{2} + \frac{s}{2N_1} + 2N_1 \log_q(\delta))$, but the error incurred by allowing values greater than 1 seems to be small for most relevant parameter choices.

The cliff height is then

$$\log_q(\|b_m^*\|) = \log_q(\|b_{r_1}^*\|) - 2N_1\log_q(\eta) \approx \frac{1}{2} + \frac{s}{2N_1} - 2N_1\log_q(\delta). \tag{22}$$

where the approximation is valid for large N_1.