Choosing NTRUEncrypt Parameters in Light of Combined Lattice Reduction and MITM Approaches

P. Hirschhorn[1], J. Hoffstein[2], N. Howgrave-Graham[3], W. Whyte[3]

[1] Wellesley College, [2] Brown University, [3] NTRU Cryptosystems, Inc.

Abstract. We present the new NTRUEncrypt parameter generation algorithm, which is designed to be secure in light of recent attacks that combine lattice reduction and meet-in-the-middle (MITM) techniques. The parameters generated from our algorithm have been submitted to several standard bodies and are presented at the end of the paper.

1 Introduction

Recent research [12] has demonstrated that the NTRUEncrypt parameter sets described in [11] do not provide the claimed level of security (for example, the parameter sets proposed in [11] for 80-bit security only provide about 64 bits of security against the attack of [12])[1]. This paper proposes new parameter sets that are secure against the attacks of [12]. In addition, we present the algorithm used to obtain those parameter sets. Some public key cryptosystems are known to be vulnerable to maliciously chosen parameters [20]; the publication of a parameter generation algorithm greatly reduces the risk that parameters have been "cooked".

The basic framework of NTRUEncrypt allows for great variation in the types of parameter sets generated. For example, polynomials used in the system may be binary, trinary, or "product-form" [8]. This paper focuses on parameter sets for trinary polynomials, as opposed to the binary polynomials of [11], because they generally offer a better trade-off between security and efficiency[2]. We make

[1] The security function used in this paper is simply the processor time required to break an instance of the algorithm. Memory is assumed to be free to the attack, although in fact the attack of [12] requires substantial amounts of memory. This approach is consistent with that recommended by the ASC X9 standard committee.

[2] Previous papers used binary parameters mainly to keep the parameter q at 256 or less, as this made the calculations more suitable for an 8-bit microcontroller; we do not currently believe it is realistic to keep q down to this level, and as such the advantages of trinary polynomials outweigh the advantages of binary polynomials. The "product-form" polynomials of [8] promise to be more efficient even than the trinary polynomials, but the analysis of the security of product-form polynomials against the attack of [12] is considerably more complicated than the analysis of trinary or binary and is not yet complete.

various other choices to restrict the space that parameters are chosen from. These choices are explained and motivated in Section 1.1.

The original contributions of this paper include:

- Explain in detail the parameter generation algorithm (because its publication helps justify that there are no "backdoors" in the chosen NTRU parameters)
- Explicitly state any underlying assumptions we use to find optimal parameters (because it is important to understand which parts of the NTRU parameters rely on provable statements and which rely on reasonable assumptions)
- Provide an exact calculation of the probability p_s that a correct guess in the meet-in-the-middle stage of the attack will be recognized as such (this was only experimentally calculated in [12])
- Provide exact calculations of various other quantities involved in the hybrid attack when applied to trinary parameters. This includes a mathematically based formula for the probability of a decryption failure with trinary polynomials
- State the standardized NTRU parameters, and their estimated security.

The algorithm has been fully implemented in an object-oriented fashion in C++. The resulting parameter sets are given in Section 7, and a discussion of future issues is given in Section 8.

NTRU parameters are also dependent on the expected running time of lattice reduction algorithms, which can be modeled by extrapolating data achieved in practice. We state the NTRU parameter generation algorithm generically, i.e. the extrapolation line is given as input to the algorithm. When calculating explicit parameter sets we use specific extrapolation lines, which are justified in Appendix A. The accuracy of these extrapolation lines is stated as an explicit assumption, since they have been generated from extensive experimentation with practical lattice reduction techniques. The extrapolation lines in this paper have been obtained using a novel lattice reduction method known as *isodual lattice reduction*, which appears to give considerably better results in practice than previous lattice reduction methods. The isodual lattice reduction method will be the subject of a separate paper.

1.1 NTRU background

Since we are describing a parameter generation function we must first define what we mean by NTRUEncrypt parameters. Let $\mathcal{R}$ denote the ring $\mathbb{Z}[X]/(X^N - 1, q)$, for some suitable integers q, N. Let $\mathcal{T}_{d_1,d_2}$ denote the set of trinary elements of $\mathcal{R}$ with d_1 entries equal to one, d_2 entries equal to minus one, and the remaining $N - d_1 - d_2$ equal to zero. Let $a \in_R A$ denote the process of picking an element a from the set A uniformly at random.

The NTRUEncrypt parameters, as described in [6], are integers (q, p, N, d_f, d_g, d_r) where the private key is $F \in_R \mathcal{T}_{d_f,d_f}$[3] and the public key is $h = g/f$ in

[3] We take $d_1 = d_2$ here because, for a given value of $d_1 + d_2$, taking $d_1 = d_2$ maximizes the work for an attacker, and because the efficiency depends only on $d_1 + d_2$, not on the individual values of d_1 and d_2.

the ring $\mathcal{R}$ where $g \in_R \mathcal{T}_{d_g+1,d_g}$[4] and $f = 1 + pF$ is invertible for most choices of F. N is necessarily a prime [5], and p is necessarily coprime to q. For any value of q, we denote the factors of q by $\{q_i\}$ and require each q_i to have order $\geq (N-1)/2 \bmod N$ to avoid attacks based on the factorization of h (or other polynomials) in the ring. This eliminates some values of N from consideration. We denote by $\mathcal{P}$ the set of allowable primes for N.

The encryption primitive forms the ciphertext $e = m' + r * h$ in $\mathcal{R}$, where m' and $r \in_R \mathcal{T}_{d_r,d_r}$ are derived from the message m with a randomized padding scheme [9–11]. We can write this primitive as a trapdoor one-way function (OWF), which is made more precise in [10, 11].

$$OWF(m', r) = m' + r * h$$

The NTRUEncrypt cryptosystem can alternatively be seen as a lattice based cryptosystem [15] in the lattice generated by the row of the following $(2N)\times(2N)$ matrix.

$$\mathcal{L}_{NTRU} = \begin{pmatrix} qI_N & 0_N \\ H & I_N \end{pmatrix},$$

where H denotes a $(N) \times (N)$ circulant matrix generated from h, i.e. $H_{i,j} = h_{i-j \bmod N}$. This lattice is also useful in cryptanalyzing NTRUEncrypt.

For all the parameter sets in this paper we will use the values $q = 2048$, $p = 3$, and $d_r = d_f$, $d_g = \lfloor N/3 \rfloor$[5]. Thus the NTRUEncrypt parameter family we are concerned with can be parametrized by just two parameters: N and d_f.

1.2 Attacks on NTRUEncrypt

At a high level there are two distinct but related attacks known on NTRUEncrypt: an attack based on knowledge of valid (plaintext,ciphertext) pairs which cause decryption failures [10], and an attack based on reversing the one-way function used for key generation and encryption [12].

The notion of decryption failures is slightly uncommon in cryptography, namely that there is a small probability p_{dec} that a valid message can fail to decrypt properly. If an attacker is aware of when this happens in NTRUEncrypt it has been noticed that they can mount an attack on the cryptosystem to recover the private key [10]. Thus in designing parameters for a security level k we ensure that this probability is less than 2^{-k} for a randomly chosen preimage (m', r) of the NTRUEncrypt OWF. The encryption scheme used [11] then provides random (m', r), allowing a proof of security in the random oracle model.

[4] g is unbalanced in this way to increase the chance that g will be invertible in $\mathcal{R}$.

[5] $q = 2048$ is the only value of q for which we have enough lattice reduction experiments to reliably extrapolate lattice reduction times. Further lattice experiments are underway at the moment. Taking q to be a power of 2 allows for efficient reduction $\bmod q$. We take $d_r = d_f$ for convenience, although the result of this is that it is very slightly easier to recover plaintext from ciphertext than private keys from public keys. We take $d_g = \lfloor N/3 \rfloor$ because the thickness of g does not affect efficiency and for security reasons it is sensible to take it to be as thick as possible.

Remaining at a high level, and recalling that our NTRUEncrypt parameter family is parametrized only by N and d_f, the consequence of requiring that decryption failures occur with small probability is that d_f cannot get too large compared to N.

Conversely the attacks based on reversing the one-way function used for key generation and encryption work better when d_f is small compared to N. The challenge facing parameter generation is thus to balance the size of d_f compared to N, whilst keeping N relatively small for efficiency.

The best known attack on reversing the NTRUEncrypt OWF is a combination of the two previously known attacks on NTRUEncrypt, namely Odlykzo's MITM attack, and standard lattice reduction. It is shown in [12] that these can be combined in to one (more efficient) hybrid attack.

The attack consists of two consecutive stages: the first stage is to reduce the initial rows of the public NTRU lattice basis $\mathcal{L}_{NTRU}$ with the best known lattice reduction scheme, and the second stage is to store partial key guesses in boxes dependent on the output of the first stage. The attack is successful if two partial key guesses that collide in the same box can be combined to recover the entire private key.

This paper is written from the perspective of justifying the new parameter generation algorithm which is secure in light of this hybrid attack[6], not from the perspective of explaining the hybrid attack further (the reader is referred to [12] for this). However, to help the reader, informative comments about the attack are embedded in to the paper whenever they are applicable.

2 An overview of parameter generation

2.1 The criteria for valid parameters

As explained in Sections 1.1 and 1.2 the challenge of generating NTRUEncrypt parameters suitable for a security level k is in choosing (N, d_f) such that:

1. The probability of decryption failures, P_{failDec}, is at most 2^{-k} for randomly chosen preimages of the NTRU OWF, *and*
2. The expected work to recover the private key f from the public key h, W_{key}, is at least 2^k, *and*
3. The expected work to recover a plaintext m from the ciphertext c, W_{msg}, is at least 2^k.

There is an implicit assumption here, namely:

Assumption 1 *If decryption failures occur with probability less than 2^{-k} then there is no attack with expected work less than 2^k which exploits the phenomenon of decryption failures.*

[6] And the decryption failure attacks.

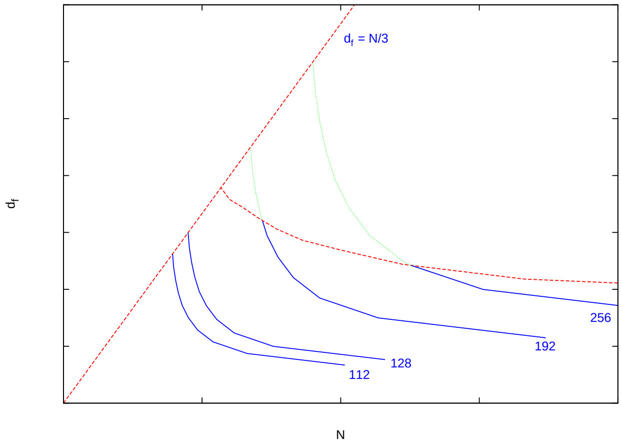
One possible line of attack to circumvent Assumption 1 is to find weaknesses in the hash functions such that one can sample preimages (m', r) to the NTRUEncrypt OWF which have decryption failures with a higher probability than randomly chosen preimages. Whilst such an avenue is theoretically possible it seems a daunting task for an attacker, and would likely require an unfeasibly large number of calls to the decryption oracle. We also note that since the hash functions take the public key as input, an attacker will with high probability need to find a separate set of preimages for each key under attack, preventing them from leveraging a single decryption-failure-causing preimage to attack multiple keys.

We also remark that there has been some work done [16] on linking requirements 2 and 3 above, i.e. in showing that if one has an oracle that can recover NTRUEncrypt messages, then it can be used to recover the NTRUEncrypt private key [16]. Such reductions are typically highly dependent on the structure of the key and OWF preimages so typically do not form tight arguments. In the proposed NTRUEncrypt parameters we have not looked for a provable correspondence between message and key recovery. Instead we independently consider the problem of message recovery in Section 6.

2.2 The space of valid parameters

The space of valid (N, d_f) for $q = 2048$, $p = 3$, $d_r = d_f$, $d_g = \lfloor N/3 \rfloor$ is depicted in the Figure 1.

Fig. 1. k-isopleths for d_f vs. N



The curved solid lines denote "k-isopleths"; parameters (N, d_f) for which W_{msg}, the expected work of the hybrid attack is 2^k. The security levels shown corresponds to $k = 112, 128, 192$ and 256 bits of security. Notice that d_f is necessarily at most $\lfloor N/3 \rfloor$ since F is a trinary vector, and this constraint is shown by the (leftmost) straight dashed line. The other constraint on d_f is

$$W_{\text{msg}} \leq P_{\text{decFail}}^{-1} \,.$$

The (central) curved dashed line corresponds to $W_{\text{msg}} = P_{\text{decFail}}^{-1}$. Valid NTRUEncrypt parameters (N, d_f) are below both of these lines.

The exact location of the isopleths of the expected work of the hybrid attack depends on how we are measuring this work. In this report we have two ways of measuring the hybrid cost:

- a *conservative* estimate (which involves being conservative about both lattice running times, and conservative about the expected running time of the MITM component).
- a *current* estimate (which involves more realistic estimates of how quickly the attack can be mounted, with our current knowledge).

Once the isopleths of the expected work of the hybrid attack are fixed, one can apply a "cost" function to each of the (N, d_f) on a k-isopleth and potentially find an "optimal" one. In this report we consider three ways of measuring the cost of a parameter set:

- a *space* metric (trying to optimize the size of the ciphertexts and public key),
- a *speed* metric (trying to optimize the time to perform decryption), and
- a *trade-off* metric (combining the two other metrics).

2.3 The algorithm

The parameter generation algorithm takes as input a security parameter k and outputs the parameter set corresponding to that security level. At a high level it can simply be described as the following: (a) pick a way to measure cost and a way to measure security, (b) find an initial (N, d_f) on the dashed line which corresponds to a security level of k, and (c) traverse the k-isopleth until the cost metric is optimized. This is made more precise in Algorithm 1.

3 Decryption failure probability

The decryption failure probability can be conservatively bounded by the probability that one or more coefficients of $r * g + F * m$ has an absolute value greater than $c = (q - 2)/(2p)$.

For trinary polynomials chosen subject to the constraints in this paper, this probability is

$$p_{dec} = N * \texttt{erfc}(c/(\sigma\sqrt{2N})), \tag{1}$$

Algorithm 1 NTRUEncrypt Parameter Generation

```
1: $i \leftarrow 1$ {The variable $i$ is used to index the set of acceptable primes $\mathcal{P}$}
2: $i^* \leftarrow 0$ {This will become the first index which can achieve the required security}
3: repeat
4:    $N \leftarrow \mathcal{P}_i$
5:    $d_f \leftarrow \lfloor N/3 \rfloor$ {We will try each $d_f$ from $\lfloor N/3 \rfloor$ down to 1}
6:    repeat
7:       $k_1 \leftarrow$ hybridSecurityEstimate$(N, d_f)$ {This is dependent on the chosen security metric}
8:       $k_2 \leftarrow \log_2$(decryptionFailureProb$(N, d_f)$)
9:       if ($k_1 \geq k$ and $k_2 < -k$) then
10:         $(i^*, d_f^*) \leftarrow (i, d_f)$ {Record the first acceptable index $i$ and the value of $d_f$}
11:      end if
12:      $d_f \leftarrow d_f - 1$
13:   until ($i^* > 0$ or $d_f < 1$)
14:   $i \leftarrow i + 1$
15: until $i^* > 0$
16: $c^* \leftarrow$ cost$(\mathcal{P}_{i^*}, d_f^*)$ {This is dependent on the chosen cost metric}
17: while an increase in $N$ can potentially lower the cost do
18:   $N \leftarrow \mathcal{P}_i$
19:   $d_f \leftarrow d_f^*$ {Note that when $N$ increases the cost must be worse for all $d_f \geq d_f^*$,
      and that the decryption probability is decreased both by an increase in $N$ and
      a decrease in $d_f$}
20:   repeat
21:      $k_1 \leftarrow$ hybridSecurityEstimate$(N, d_f)$
22:      $c \leftarrow$ cost$(N, d_f)$
23:      if ($k_1 \geq k$ and $c < c^*$) then
24:         $(c^*, i^*, d_f^*) \leftarrow (c, i, d_f)$ {Record the improvement in cost and the corresponding $i, d_f$}
25:      end if
26:      $d_f \leftarrow d_f - 1$
27:   until $d_f < 0$
28:   $i \leftarrow i + 1$
29: end while
30: return $(\mathcal{P}_{i^*}, d_f^*)$
```

- Note that line 17 is not specified precisely because it depends on the cost() function used (see Section 4 for a discussion of the possible cost metrics and end-conditions).
- Note that the call to decryptionFailureProb() on line 8 simply uses Equation 1, and the calls to hybridSecurityEstimate() use Algorithm 2.

where $\sigma^2 = 4(d_r + d_f)/(3N)$.

We now justify this statement. In the following calculation we will make the following explicit assumption:

Assumption 2 *The coefficients of r are independent random variables taking the value 1 with probability d_r/N, -1 with probability d_r/N and 0 with probability $(N-2d_r)/N$. We make corresponding assumptions about g, d_g, F, d_f and $m', d_{m'}$.*

In order for decryption to succeed, the coefficients of

$$a = p * (r * g + m' * F) + m.$$

must have absolute value less than $q/2$. In fact, it would suffice to know that these coefficients lay in an interval of length less than q in order to do first a translation, then a decryption. Shortly after NTRU was first introduced this second method was viewed as preferential, as it increased the probability of successful decryption. However, theoretically predicting the likelihood of decryption failure by this method was a particularly unwieldy problem, due to the fact that the sizes of the maximum and minimum coefficients were not independent random variables.

The somewhat weaker question of estimating from above the probability

$$p_{dec}(c) = \text{Prob}\left(\text{a given coefficient of } r * g + m' * F \text{ has absolute value } \geq c\right) \tag{2}$$

can, however, be analyzed quite accurately by a simple application of the central limit theorem, making the assumption described above.

If X_j denotes a coefficient of $r * g + m' * F$, then X_j is a sum of N terms, that is,

$$X_j = \sum_{i=1}^{N} (y_i + z_i),$$

where each $y_i = r_k g_l$ and $z_i = m_s F_t$ for some k, l, s, t. It is then easily checked that

$$\sigma^2 = E(X_j^2) = \sum_{i=1}^{N} \left(E(y_i^2) + E(z_i^2)\right) = \frac{4d_r d_g + 4d_f d_{m'}}{N}.$$

Assuming that N is large, we apply the central limit theorem to X_j, (normalized to have variance equal to 1). Applying the theorem twice, to account for the negative or positive extremes of X_j, we find that

$$\text{Prob}\left(|X_j| \geq C\sigma\right) < \frac{2}{\sqrt{2\pi}} \int_C^\infty e^{-x^2/2} dx.$$

Writing $c = C\sigma$ we have

$$\text{Prob}\left(|X_j| \geq c\right) < \frac{2}{\sqrt{2\pi}} \int_{c/\sigma}^\infty e^{-x^2/2} dx.$$

Translating this into the notation of the complementary error function, we get

$$\text{Prob}\left(|X_j| \geq c\right) < \texttt{erfc}(c/(\sigma\sqrt{2})),$$

with σ as above.

The decryption failure probability can be conservatively bounded by the probability that one or more coefficients of $r * g + F * m$ has an absolute value greater than $c = (q-2)/(2p)$. Thus, repeating the experiment of selecting a coefficient N times, we finally have the probability of decryption failure bounded above by

$$p_{dec}(c) = N\texttt{erfc}(c/(\sigma\sqrt{2})),$$

with $c = (q-2)/(2p)$. For trinary polynomials chosen subject to the constraints in this paper, $\sigma^2 = 4(d_r + d_f)/3$. Experiments with on the order of 10^9 polynomials, with different values for the d's, have shown that this model gives an accurate, while conservative, description of reality.

4 Cost functions

As explained in Section 2.2 there are three cost metrics of interest:

The space metric The space metric corresponds to the bit-size of the public key and ciphertexts, i.e. $\text{cost}_{\text{space}} = N \log_2 q$. Since, for constant q, this is independent of d_f, one does not have to traverse the k-isopleth looking to minimize it; it is simply minimized for the least N on the k-isopleth.

The speed metric The speed metric is derived from the convolution times for a parameter set, i.e. $\text{cost}_{\text{speed}} = Nd_f$.
When trying to minimize this cost notice that an increase in N is offset by a decrease in d_f, thus if N^*, d_f^* corresponds to the current minima on the k-isopleth then one can stop searching the k-isopleth when N increases beyond $N^* d_f^*/(d_f^* - 1)$ without a drop in d_f.

The trade-off metric In particular environments neither the space metric nor the speed metric may be the natural way to measure cost, but some other metric. We introduce a trade-off metric: $\text{cost}_{\text{trade-off}} = \text{cost}_{\text{space}}^2 \times \text{cost}_{\text{speed}}$, which does not have an obvious intrinsic interest, but allows us to show how to handle different metrics.
The criteria for knowing when to stop searching the k-isopleth for this metric can be the same as the speed metric.

5 Hybrid Security

5.1 Overview

The hybrid security function aims to find optimal attack parameters, i.e. parameters such the maximum of lattice security and MITM security is minimized over all attack parameters.

As explained in [12], the hybrid security of a parameter set (N, d_f) is the minimum security over all attack strategies on that parameter set. An attack strategy is defined by the following three values:

- an integer y_2, $N \leq y_2 \leq 2N$ which corresponds to the number of initial rows of the lattice $\mathcal{L}_{NTRU}$ which will be reduced
- a real number α, which roughly corresponds to the quality of the reduced part of the lattice; a larger α corresponds to a better reduced lattice (which takes more time to achieve).
- an integer c which corresponds to the expected number of ± 1's in the last $2N - y_2$ entries of F which we will mount the MITM attack on (note that these entries affect the last $2N - y_2$ rows of $\mathcal{L}_{NTRU}$). By definition, $0 \leq c \leq d_f$.

In performing our analysis we have assumed that the number of $+1$'s and -1's in the last $2N - y_2$ rows is the same. For clarity we make this assumption explicit:

Assumption 3 *An attacker cannot do substantially better by assuming that the number of* 1 *coefficients in the last* $2N - y_2$ *rows is different from the number of* -1 *coefficients.*

A simple way to enumerate the space and hence find the optimal attack strategy is given in Algorithm 2.

Algorithm 2 Optimal Attack Strategy and Hybrid Security Estimation

```
 1: for $y_2 = N$ to $2N$ do
 2:   for $c = 0$ to $d_f$ do
 3:     Find $\alpha$ such that:
        latticeRunningTime$(y_2, \alpha)$ − MITMRunningTime$(y_2, c, \alpha) \approx 0$.
 4:     $w \leftarrow$ max(latticeRunningTime$(y_2, \alpha)$, MITMRunningTime$(y_2, c, \alpha)$)
 5:     if ($w^*$ is undefined or $w < w^*$) then
 6:       $(w^*, y_2^*, c^*, \alpha^*) \leftarrow (w, y_2, c, \alpha)$ {Record the improved attack strategy, and
          implied attacker work}
 7:     end if
 8:   end for
 9: end for
10: return the optimal attack strategy $(y_2^*, c^*, \alpha^*)$ and the estimated work $w^*$
```

- Notice that the call to latticeRunningTime() is independent of c and simply returns Equation 3. The call to MITMRunningTime() returns $t\mathcal{N}/p_{split}$ where these quantities are defined as in Section 5.3.
- The calculation of α in step 3 can be done by standard root finding techniques.

5.2 Lattice Security

In order to find optimal attack parameters, we need to be able to estimate the work it takes to reduce the first y_2 rows of the lattice $\mathcal{L}_{NTRU}$ with "quality" α.

In Appendix A we discuss what it means for lattice to have "quality" α, and we present the lattice experiments we have conducted to approximate this time. To extrapolate beyond the end of the experimental data we model running time as exponentially dependent on the GSA slope as defined in [19], and cubically dependent on the dimension[7]. This model gives the estimates of running time as 2^w where

$$w = \frac{2m(y_2 - N)}{(1-\alpha)^2} + 3\log\frac{2(y_2 - N)}{1-\alpha} + c \qquad (3)$$

for some constant m, c.

Extrapolating lattice reduction times is far from a science, but the data seems to support, with reasonable assurance, that practical lattice reduction times exceed the above formula with $m = 0.45$, $c = -110$.

As mentioned in Section 2.2 we allow the parameters sets to be generated with either current security assurances, or conservative security assurances. For the conservative security assurance we use the constants $m = 0.2$, $c = -50$, which hopefully give ample room for improvements in lattice reduction, without requiring that the NTRUEncrypt parameter sets change. We make this assumption explicit below:

Assumption 4 *We assume lattice reduction of the first y_2 elements of the lattice $\mathcal{L}_{NTRU}$ with quality α takes time at least 2^w where w is given by Equation 3, with $m = 0.2$, $c = -50$.*

5.3 MITM Security

In order to find optimal attack parameters we need to be able to estimate the expected work it takes to perform the MITM attack, for a given y_2, α and c.

The following events need to happen for a successful MITM attack:

- The c value must correctly hold how many 1s and -1s are in the last $2N - y_2$ entries of F. The probability of this event is called p_{split}.
- The attacker must enumerate through a large number of guesses of "halves" of the end of F.
- For each one he must apply a CVP reduction algorithm.
- A pair of half-guesses must collide after the CVP algorithm. This probability is referred to as p_s in [12].

We now go through the expected work for each of these events. If a necessary event happens only with a probability p then the expected work is divided by p.

Let $\binom{n}{r_1, r_2}$ denote a "trinomial" quantity, i.e. the number of ways of choosing r_1 positions of one kind, and r_2 positions of another kind in a vector of length n. It is simple to confirm the following relationship between trinomial and binomial quantities: $\binom{n}{r_1, r_2} = \binom{n}{r_1}\binom{n-r_1}{r_2}$.

[7] This cubic dependency is heuristic but does not greatly affect the results.

The probability p_{split} The probability that the number of 1s and -1s in the last $2N - y_2$ entries of F is equal to c is given by

$$p_{\text{split}} = \binom{y_2 - N}{d_f - c, d_f - c}\binom{2N - y_2}{c, c}\binom{N}{d_f, d_f}^{-1}$$

If the NTRUEncrypt public key was $h = g/F$ then any of the rotations of F would be suitable for the MITM attack, so this probability should be increased to

$$p'_{\text{split}} = 1 - (1 - p_{\text{split}})^N,$$

assuming the N rotations behave like independent trials.

In the case when the NTRUEncrypt public key is $h = g/f$ then we do not see a way to exploit the cyclic symmetry, so when assessing the MITM work in the *current security* case we will be using the probability p_{split}. However, when assessing the MITM work in the *conservative security* case we will assume that the rotations can still somehow be exploited and use the probability p'_{split}.

The probability p_s In [12] the probability p_s was experimentally calculated, whereas for parameter generation we must calculate this probability mathematically. To make this possible we make the following assumption (which very closely models reality).

Assumption 5 *We assume the orthonormal matrix Y defined in [12] (which is an output of the lattice reduction stage) is suitably random so that the error vector $(g|f_1)Y$ can be modeled by a normal distribution with mean zero, and variance $\sigma^2 = |g|^2 + |f_1|^2 = 2d_g + 2(d_f - c)$.*

As explained in [12] the probability p_s denotes the probability that an "error vector" e is such that $\text{Babai}_B(v) = \text{Babai}_B(v + e)$ where $\text{Babai}_B(v)$ denotes applying Babai's nearest plane CVP algorithm to the point v with respect to the basis B, i.e. the addition of the error vector e does not change the returned close lattice vector.

In NTRUEncrypt the error vector e results from the multiplication of a trinary vector (of norm-squared $2d_g+2(d_f-c)$) with an orthonormal matrix, Y, resulting from lattice reduction. Under the assumption that Y is "suitably random", we can model the error vector e as (y_2-y_1) coefficients drawn independently from a normal distribution with mean 0 and variance $\sigma^2 = 2d_g + 2(d_f - c)$. This model fits extremely closely with experimental results from 10^9 separate convolution calculations.

Let v be a vector of length $y_2 - y_1$ with coefficients satisfying $0 \le v_i \le q^{\alpha+i(1-\alpha)/(y_2-y_1)}$, i.e. v is weakly reduced with respect to a basis that satisfies the GSA. In this case p_s is the probability that the vector $w = v + e$ satisfies the conditions $0 \le w_i \le q^{\alpha+i(1-\alpha)/(y_2-y_1)}$ for every i. We now show how to calculate p_s exactly, assuming the above model of the error vector e.

Let $D = q^\beta$ for some $\alpha \le \beta \le 1$. We assume any given coefficient is uniform in the range $[0, D)$ and is subject to the addition of an error term drawn from a normal distribution with variance σ^2 and mean 0. For any given $x \in [0, D)$ the probability that this sum is no longer in the range $[0, D)$ – in other words, the probability that a particular coefficient will be reduced incorrectly – is given by:

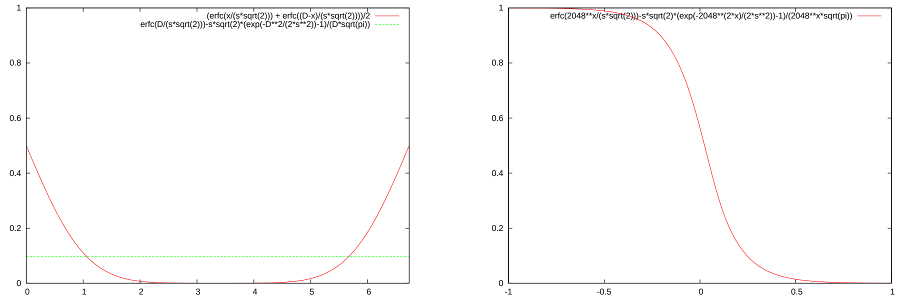
$$\begin{aligned} f_{D,\sigma}(x) &= \frac{1}{2}\left(\operatorname{erfc}\left(\frac{x}{\sigma\sqrt{2}}\right) + \operatorname{erfc}\left(\frac{D-x}{\sigma\sqrt{2}}\right)\right) \\ &= 1 - \frac{1}{2}\left(\operatorname{erf}\left(\frac{x}{\sigma\sqrt{2}}\right) + \operatorname{erf}\left(\frac{D-x}{\sigma\sqrt{2}}\right)\right) \end{aligned}$$

See Figure 2(a) for an example of this function. Note that, so long as $\sigma^2 < D$, we have $f_{D,\sigma}(0) = 0.5$ and $f_{D,\sigma}(D) = 0.5$ as the values of e_i will be positive or negative with probability 0.5. The expected value of f when x is chosen uniformly from $[0, D)$ is given by:

$$\overline{f_{D,\sigma}} = \frac{1}{D}\int_0^D f_{D,\sigma}(x)dx.$$

This value is also plotted in Figure 2(a).

Fig. 2. (a) $f_{D,\sigma}(x)$ and $\overline{f_{D,\sigma}}$ when $D = 2048^{1/4}$, $\sigma^2 = 2/3$, $0 \le x \le D$; (b) $\overline{f_{q^\beta,\sigma}}$ when $q = 2048$, $\sigma^2 = 2/3$, $-1 \le \beta \le 1$



Now we can analyze the behavior of $\overline{f_{D,\sigma}}$ with D. From well known properties about the error function $\operatorname{erf}()$ we know that:

$$\begin{aligned} \int_0^D \operatorname{erf}\left(\frac{x}{\sigma\sqrt{2}}\right) dx &= \sigma\sqrt{2}\int_0^{\frac{D}{\sigma\sqrt{2}}} \operatorname{erf}(y)\,dy \\ &= \sigma\sqrt{2}\left[y \operatorname{erf}(y) + \frac{e^{-y^2}}{\sqrt{\pi}}\right]_0^{\frac{D}{\sigma\sqrt{2}}} \\ &= D \operatorname{erf}\left(\frac{D}{\sigma\sqrt{2}}\right) + \frac{\sigma\sqrt{2}}{\sqrt{\pi}}\left(e^{-\frac{D^2}{2\sigma^2}} - 1\right) \end{aligned}$$

and

$$\int_0^D \operatorname{erf}\left(\frac{D-x}{\sigma\sqrt{2}}\right) dx = \int_0^D \operatorname{erf}\left(\frac{x}{\sigma\sqrt{2}}\right) dx.$$

Thus we have

$$\overline{f_{D,\sigma}} = \operatorname{erfc}\left(\frac{D}{\sigma\sqrt{2}}\right) - \frac{\sigma\sqrt{2}}{D\sqrt{\pi}}\left(e^{-\frac{D^2}{2\sigma^2}} - 1\right).$$

This function is sketched in Figure 2(b) with $D = q^{\beta}$, $-1 \leq \beta \leq 1$.

Now we can calculate $p_s(y_2, \alpha, q, \sigma)$ which is the probability that none of the coefficients in the middle $y_2 - y_1$ coefficients of γ are reduced incorrectly (note that y_1 is related to y_2 and α by (4)):

$$\begin{aligned} p_s &= \left(1 - \frac{2}{3q}\right)^{y_1} \prod_{i=0}^{y_2-y_1} \left(1 - \overline{f_{q^{\frac{\alpha(y_2-y_1)+i(1-\alpha)}{y_2-y_1}},\sigma}}\right) \\ &= \left(1 - \frac{2}{3q}\right)^{\frac{2N-y_2(1+\alpha)}{1-\alpha}} \prod_{i=0}^{\frac{2(y_2-N)}{1-\alpha}} \left(1 - \overline{f_{q^{\frac{2\alpha(y_2-N)+i(1-\alpha)^2}{2(y_2-N)}},\sigma}}\right) \end{aligned}$$

The time for the CVP algorithm Each step in the MITM search phase involves a reduction against a $(y_2) \times (y_2)$ matrix. In this paper we assume the reduction is performed by Babai's method [1]. Babai's reduction is experimentally found to take about $t = y_2^2/2^{1.06}$ operations where bit operations are defined as in [14].

For conservative estimates, we assume that since the MITM phase involves multiple reductions against the same reduced matrix, there will be some optimization involving precomputation that reduces the running time by a factor of y_2 to $t' = y_2/2^{1.06}$.

We note that there may be other means of carrying out this weak-CVP reduction phase. Babai's reduction is likely to be the most efficient when measured purely in terms of the time for the reduction. However, it is conceivable that a slower reduction algorithm exists which results in higher values for p_s, and that this algorithm might give better results for an attacker than the current approach.

Assumption 6 *We assume that Babai's nearest plane algorithm is the most effective CVP approach to use in the hybrid attack (in terms of collision probability divided by time).*

We note that contradicting this assumption does not necessarily mean that the proposed NTRUEncrypt parameters are weak, since conservative security estimates have been made in many other areas. However it is an interesting open question to know whether or not there is a more efficient CVP approach to be used in the hybrid attack.

The expected number of half-guesses required It is explained in [12] that the hybrid attack works by choosing a linear combination of the last $2N - y_2$ rows of $\mathcal{L}_{NTRU}$ with $c/2$ 1s and $c/2$ -1s. The number of such combinations is given by

$$\mathcal{N}_0 = \binom{2N - y_2}{c/2, c/2} \binom{c}{c/2, c/2}^{-1}.$$

However since we require the two halves to collide after the CVP stage the probability of picking such a "good" combination is only $p_s \mathcal{N}_0^{-1}$.

Moreover in [12] it is argued that, by the birthday paradox, one should expect to try $\sqrt{p_s \binom{c}{c/2, c/2}}$ samples before finding two halves that actually match. Thus the number of trials before a collision will occur for the *current security* is approximated by

$$\mathcal{N} = \binom{2N - y_2}{c/2, c/2} \left(\binom{c}{c/2, c/2} p_s \right)^{-1/2}$$

In estimating the number of trials before collision with occur with *conservative security* we assume that generalized birthday methods can be applied so just one[8] "good" combination is sufficient to find a collision, i.e. we use the estimate $\mathcal{N}' = \mathcal{N}_0 / p_s$. This conservative assumption may be close to reality as indicated by the work in [13].

6 Message recovery

One can estimate the hardness of message recovery as opposed to key recovery in a similar way to the above. The ciphertext point $(c, 0)$ is only $(m', -r)$ away from a lattice point of $\mathcal{L}_{NTRU}$. In our family of NTRU parameters we know $d_r = d_f$, but m' is a random trinary vector, with an un-fixed number of 0s, 1s and -1s. If m' is very sparse for example, then message recovery could be easier than key recovery. We also note that the value of $m'(1)$ is leaked by each ciphertext since $c(1) = m'(1) + r(1)h(1) \bmod q$.

Clearly an encrypter cannot leak the private key in a public-key encryption scheme (since they know no more information than an attacker), but they might leak their message m. However we observe that the encrypter does see m', r, so she can re-encrypt if she wants. We note that the probability of re-encryption depends on the parameter set, and parameters may be rejected if this probability is too high for comfort.

In this report we make the following simplifying assumption:

Assumption 7 *An encrypter that re-encrypts whenever the number of* 1*s or* -1*s in* m' *falls below* d_f *does not make message recovery fall below the required security level.*

[8] Or more realistically a small constant number of good combinations

When comparing the message recovery problem to the key recovery problem there are two factors to take in to consideration: one that is good for an attacker, and one that is bad. Firstly if an attacker knows m' only has d_f 1s and -1s, the message recovery lattice problem is actually slightly easier than the key recovery problem since p_s will be larger in this case (due to the fact that g is thicker than m'). However the second factor is that m' is unknown to an attacker, so the probability of a sparse m' should be factored in to an attackers strategy. We note that the probability of a random trinary N-vector having d_m ones and d_m minus ones is given by $\binom{N}{d_m, d_m}$.

To garner belief in Assumption 7 we compared the expected p_s for message security to the p_s for key security (in the case of ciphertexts satisfying $m'(1) = 0$) for each of the optimal attack parameters given in Section 7. The expected p_s probability for the message recovery problem was improved by at most 0.1 bits from the key recovery p_s probability, which suggests that message recovery is very closely tied to key security, and hence Assumption 7 is realistic.

7 Parameter Sets

The results of running the parameter generation algorithm under conservative assumptions about the strength of lattice attacks and MITM attacks is given in Table 1. For each security level k we give the optimized parameter sets (N, d_f) for each of the three cost metrics.

We also assess the strength, k', of parameter sets under current assumptions about the strength of lattice attacks, and MITM attacks, to show the safety margin "built-in" to the parameters.

Table 1. Standardized NTRU Parameters (conservative)

	space	trade-off	speed
k	$(N, d_f), k'$	$(N, d_f), k'$	$(N, d_f), k'$
112	$(401, 113), 154.88$	$(541, 49), 141.766$	$(659, 38), 137.861$
128	$(449, 134), 179.899$	$(613, 55), 162.385$	$(761, 42), 157.191$
192	$(677, 157), 269.93$	$(887, 81), 245.126$	$(1087, 63), 236.586$
256	$(1087, 120), 334.85$	$(1171, 106), 327.881$	$(1499, 79), 312.949$

For the reader that is interested in knowing the exact attack parameters which match lattice security and MITM security (and hence give the security level), we give this information in Table 2. In this table Y denotes the number of iterations of the MITM attack, and t denotes the bit-security per iteration (i.e. the cost of Babai's CVP algorithm).

8 Conclusions

We have introduced a family of NTRU parameters parametrized by (N, d_f) and shown how to estimate their strength. We have used this algorithm to generate parameters sets for the $k = 112$, 128, 192 and 256-bit security levels under conservative assumptions about the effectiveness of lattice reduction algorithms and the MITM attack. To highlight the safety margin we have built in to the new parameters we have also estimated how hard the parameters are to attack under more current assumptions.

A future line of work may be in expanding the size of the NTRU family of parameters to allow for even better optimized parameters. For example the family we have defined uses a fixed $q = 2048$, whereas a smaller q could reduce the bandwidth and public key-size used in the cryptosystem. To propose such parameters more lattice experiments would have to be run at lower values of q, and one would need to ensure that the decryption failure probability is still small enough.

Table 2. Optimal attack parameters

	space	trade-off	speed
k	$(N, d_f), (y_2, c, \alpha)$ $(y_1, p_s, p_{split}, Y, t)$	$(N, d_f), (y_2, c, \alpha)$ $(y_1, p_s, p_{split}, Y, t)$	$(N, d_f), (y_2, c, \alpha)$ $(y_1, p_s, p_{split}, Y, t)$
112	$(401, 113), (693, 27, 0.095)$ $(48, -45.4, -0.6, 57.7, 8.4)$	$(541, 49), (800, 15, 0.149)$ $(192, -26.9, -13.1, 63.6, 8.6)$	$(659, 38), (902, 13, 0.175)$ $(313, -21.9, -17.7, 63.7, 8.8)$
128	$(449, 134), (770, 35, 0.100)$ $(57, -49.0, -0.3, 70.2, 8.5)$	$(613, 55), (905, 17, 0.142)$ $(225, -31.5, -14.9, 72.9, 8.8)$	$(761, 42), (1026, 15, 0.183)$ $(378, -23.1, -20.9, 75.2, 8.9)$
192	$(677, 157), (1129, 45, 0.096)$ $(129, -67.4, -2.0, 113.6, 9.1)$	$(887, 81), (1294, 27, 0.143)$ $(345, -43.9, -21.9, 117.0, 9.3)$	$(1087, 63), (1464, 23, 0.175)$ $(550, -34.2, -31.9, 116.7, 9.5)$
256	$(1087, 120), (1630, 39, 0.127)$ $(386, -64.1, -24.9, 157.9, 9.6)$	$(1171, 106), (1693, 37, 0.144)$ $(474, -56.0, -28.7, 161.8, 9.7)$	$(1499, 79), (1984, 29, 0.174)$ $(809, -44.4, -47.8, 153.9, 9.9)$

References

1. L. Babai, *On Lovasz' lattice reduction and the nearest lattice point problem*, Combinatorica, 6(1): 1-13, 1986.
2. S. Cavallar, B. Dodson, A.K. Lenstra, W. Lioen, P.L. Montgomery, B. Murphy, H.J.J. te Riele, et al. Factorization of a 512-bit RSA modulus, Proceedings Eurocrypt 2000, LNCS 1807, 1–17, Springer 2000.
3. D. Coppersmith and A. Shamir, *Lattice Attack on NTRU*, Advances in Cryptology - Eurocrypt'97, Springer-Verlag
4. N. Gama and P.Q. Nguyen *Predicting Lattice Reduction*, Eurocrypt '08, pp. 31–51

5. C. Gentry, Key recovery and message attacks on NTRU-composite, *Advances in Cryptology —Eurocrypt '01*, LNCS 2045. Springer-Verlag, 2001
6. J. Hoffstein, J. Pipher, J.H. Silverman, *NTRU: A new high speed public key cryptosystem*, in Algorithmic Number Theory (ANTS III), Portland, OR, June 1998, Lecture Notes in Computer Science 1423 (J.P. Buhler, ed.), Springer-Verlag, Berlin, 1998, 267–288.
7. J. Hoffstein and J. H. Silverman. Invertibility in truncated polynomial rings. Technical report, NTRU Cryptosystems, October 1998. Report #009, version 1, available at `http://www.ntru.com`.
8. J. Hoffstein and J. H. Silverman. *Random small hamming weight products with applications to cryptography*, Discrete Applied Mathematics 130(1), 37-49 (2003)
9. N. Howgrave-Graham, P. Nguyen, D. Pointcheval, J. Proos, J. H. Silverman, A. Singer, W. Whyte, *The Impact of Decryption Failures on the Security of NTRU Encryption*, Advances in Cryptology—Crypto 2003, Lecture Notes in Computer Science 2729, Springer-Verlag, 2003, 226-246.
10. N. Howgrave-Graham, J. H. Silverman, A. Singer and W. Whyte. NAEP: Provable Security in the Presence of Decryption Failures IACR ePrint Archive, Report 2003-172, `http://eprint.iacr.org/2003/172/`
11. N. Howgrave-Graham, J.H. Silverman, W. Whyte *Choosing Parameter Sets for NTRUEncrypt with NAEP and SVES-3* CT-RSA 2005, pp. 118-135.
12. N. Howgrave-Graham, *A hybrid meet-in-the-middle and lattice reduction attack on NTRU*, pp. 150–169, CRYPTO 2007
13. A. Joux, N. Howgrave-Graham, *Generalized birthday problems applied to subset sum*, Manuscript.
14. A. Lenstra, E. Verheul *Selecting Cryptographic Key Sizes*, Journal of Cryptology 14(4), 255-293, 2001
15. D. Micciancio *Improving Lattice Based Cryptosystems Using the Hermite Normal Form*, CaLC 2001, pp. 126–145.
16. P. Mol, M. Yung *Recovering NTRU Secret Key from Inversion Oracles* PKC 2008, pp. 18–36
17. R. Rivest, A. Shamir, L. M. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, 21 (1978), 120-126.
18. RSA Laboratories, RSAES-OAEP Encryption Scheme `ftp://ftp.rsasecurity.com/pub/rsalabs/rsa_algorithm/rsa-oaep_spec.pdf`
19. C. P. Schnorr, *Lattice Reduction by Random Sampling and Birthday Methods*, STACS 2003
20. S. Vaudenay, *Hidden Collisions on DSS*, Advances in Cryptology—Crypto 1996, Lecture Notes in Computer Science 1109, Springer-Verlag, 1996, 83-88.

A Lattice Experiments

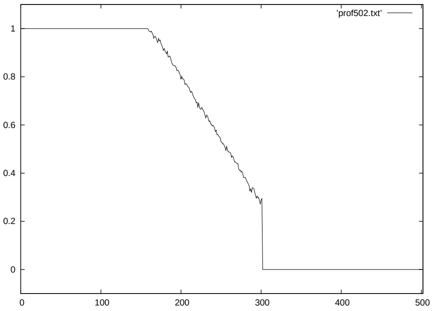
In this section we aim to give plausible estimates for the running time to reduce the first y_2 rows of the NTRU lattice $\mathcal{L}_{NTRU}$ with "quality" α.

It would be nice if there were standard ways to do this is the academic literature, but unfortunately no such work has been done. The paper [4] sounds promising but, despite the title, it does not allow us to predict lattice reduction times for a given quality of reduced basis if we, say, had computing power equivalent to 2^{80} operations.

We note that our notion of the reduction quality α is not strictly a traditional notion[9] of the quality of reduction of a lattice, but it is the most natural measure for the MITM attack.

We define the *profile* of an NTRU basis $\{b_1, b_2, \ldots, b_{2N}\}$ to be a plot of the $\log_q |b_i^*|$. As in [12] we assume that the profile after lattice reduction has taken place looks like Figure 3, that is there is an initial flat portion, followed by an almost linear reduction in the $|b_i^*|$, as predicted by the Geometric Series Assumption (GSA) first stated in [19].

Fig. 3. $\log_{197} |b_i^*|$ for $i = 1, \ldots, 502$, with $y_2 = 302$



The final $2N - y_2$ rows are shown in Figure 3 are not touched by lattice reduction (so the $\log_q |b_i^*|$ remain 0). We define y_1 to be the number of the initial "reduced" vectors satisfying $\log_q |b_i^*| = 1$. These initial y_1 vectors are also not really touched by lattice reduction, so we look on this as a $y_2 - y_1$ dimensional lattice problem. Indeed, as explained in [12], the reduced basis may be produced by extracting and reducing a $(y_2 - y_1) \times (y_2 - y_1)$ submatrix from $\mathcal{L}_{NTRU}$, and then applying some simple matrix operations.

We define $\alpha = |b_{y_2}^*|$ to be the size of the last b_i^* in the reduced portion, and we note that spending more time on lattice reduction will increase this value.

Note that, by the invariance of the determinant, and assuming the GSA, all such profiles satisfy $y_1 + (1/2)(y_2 - y_1)(1 + \alpha) = N$, so we have:

$$y_1 = \frac{2N - y_2(1 + \alpha)}{1 - \alpha}, \tag{4}$$

[9] Traditional notions typically involve the ratio of the smallest vector found to the smallest vector in the lattice, but they are closely related to our notion since such bounds generally come from bounding the last $|b_i^*|$ achieved by lattice reduction.

which implies the dimension of the lattice problem is $n = y_2 - y_1 = 2(y_2 - N)/(1 - \alpha)$.

We define the "GSA-slope" δ to be the average of $\delta_i = \log|b_i^*| - \log|b_{i+1}^*|$ for $y_1 < i \leq y_2$. Assuming the GSA we have $\delta = (1-\alpha)^2/(2(y_2 - N))$.

A reasonable way to model[10] lattice reduction running times is to expect a small polynomial dependency on the lattice dimension (i.e. n^3), and a singly exponential[11] dependency on δ^{-1}.

As part of research for this paper we have developed a new and very effective lattice reduction technique, which we refer to as "isodual lattice reduction". We will describe this technique more fully in a separate paper. The effectiveness of this technique is shown by the fact that we have achieved reduced bases in time about 2^{43} which if one had simply increased the BKZ blocksize would have needed about 2^{100} work. The isodual running times are broadly in line with the model of running times presented in this paper.

We suggest two levels of the asymptotic hardness of lattice reduction on NTRU lattices with $q = 2048$. One generated by extrapolating the (end of the) best known running times in low dimension (Equation 5), and the other (Equation 6) is proposed as a more aggressive challenge for lattice reduction:

$$\log_2 t = 0.2\delta^{-1} + 3\log n - 50 = 0.4\frac{y_2 - N}{(1-\alpha)^2} + 3\log\frac{2(y_2 - N)}{1-\alpha} - 50, \text{ or} \quad (5)$$

$$\log_2 t = 0.45\delta^{-1} + 3\log n - 110 = 0.9\frac{y_2 - N}{(1-\alpha)^2} + 3\log\frac{2(y_2 - N)}{1-\alpha} - 110. \quad (6)$$

Figure 4 shows the number of initial q-vectors which were removed for various lattice strategies, and the asymptotes with $\alpha = 0$.

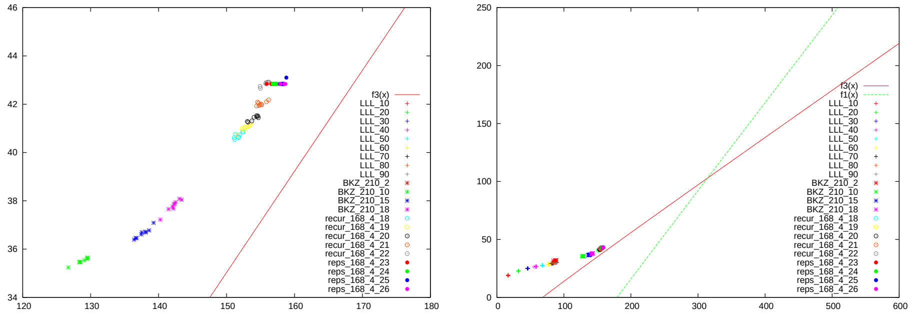


Fig. 4. Best known lattice running times

[10] The veracity of this model should be examined further, especially when $\alpha > 0$, but it does seem a reasonable model.

[11] It can be no "more secure" that this asymptotically since an NTRU-lattice can be fully searched in time q^N.