New Chosen-Ciphertext Attacks on NTRU

Nicolas Gama[1],Phong Q. Nguyen[2]

[1] École normale supérieure, DI, 45 rue d'Ulm, 75005 Paris, France
nicolas.gama@ens.fr
[2] CNRS/École normale supérieure, DI, 45 rue d'Ulm, 75005 Paris, France
http://www.di.ens.fr/~pnguyen

Abstract. We present new and efficient key-recovery chosen-ciphertext attacks on NTRUENCRYPT. Our attacks are somewhat intermediate between chosen-ciphertext attacks on NTRUENCRYPT previously published at CRYPTO '00 and CRYPTO '03. Namely, the attacks only work in the presence of decryption failures; we only submit valid ciphertexts to the decryption oracle, where the plaintexts are chosen uniformly at random; and the number of oracle queries is small. Interestingly, our attacks can also be interpreted from a provable security point of view: in practice, if one had access to a NTRUENCRYPT decryption oracle such that the parameter set allows decryption failures, then one could recover the secret key. For instance, for the initial NTRU-1998 parameter sets, the output of the decryption oracle on a single decryption failure is enough to recover the secret key.

1 Introduction

NTRU [8] is one of the fastest public-key cryptosystems known, offering both encryption (under the name NTRUENCRYPT) and digital signatures (under the name NTRUSIGN [7]) using inexpensive operations on polynomials with small coefficients. Besides efficiency, another interesting feature of NTRU compared to traditional public-key cryptosystems based on factoring or discrete logarithm is its potential resistance to quantum computers: no efficient quantum algorithm is known for the NP-hard lattice problems related to the security of NTRU. The security and insecurity of NTRU primitives has been an active research topic in the past 10 years, and NTRU is now being considered by the *IEEE P1363.1* standards [12].

While cryptanalysis has been rather successful on NTRU signatures (the basic version of NTRUSIGN has recently been broken in [14], and all the versions of its ancestor NSS were successfully attacked [4,5]), it can be argued that no significant weakness has ever been found on NTRU encryption. To date, the most dangerous attacks on NTRUENCRYPT are perhaps key-recovery chosen-ciphertext attacks. The first key-recovery chosen-ciphertext attacks were found by Jaulmes and Joux [13] at CRYPTO '00, and used few oracle queries. However, the attacks used invalid ciphertexts of very special shape, and do not seem to work for all NTRU instantiations. In particular, they can easily be thwarted by an appropriate padding scheme (as is often the case in public-key encryption), which is anyway necessary to achieve strong security notions. At CRYPTO '03, Howgrave-Graham *et al.* [11] realized that an unusual property of NTRUEncrypt known as *decryption failure* gave rise to much more powerful chosen-ciphertext attacks. Until the publication of [11] (and even [10]), all parameter sets proposed by NTRU allowed decryption failures: the ciphertext of a randomly chosen message could fail to decrypt correctly when using the NTRU decryption algorithm. Although the probability of decryption failures was small, it was significant enough (ranging from 2^{-12} to 2^{-40}) not to be ignored in practice: an attacker might realistically collect decryption failures. The most powerful chosen-ciphertext attack of [11] then allowed to attack any instantiation of NTRU, independently of the padding scheme, and using only a weak decryption oracle which asserts if a given (valid) ciphertext failed to decrypt or not. However, this attack required a large number of decryption failures (estimated to be about a million by [11]), and had not been fully implemented. In particular, the attack uses in the final stage a sophisticated algorithm by Gentry and Szydlo [5] (designed to attack the NSS signature scheme), which is polynomial time, but has to the best of our knowledge not been fully implemented.

Our results. In this paper, we present new and efficient chosen-ciphertext attacks on NTRUEncrypt. Our attacks are somewhat intermediate between the attacks of Jaulmes and Joux [13], and those of Howgrave-Graham *et al.* [11]. Like [11], the attacks are based on decryption failures and only query the decryption oracle on valid ciphertexts. However, unlike [11], we do not only ask whether a given (valid) ciphertext fails to decrypt, we ask for the full output of the NTRU decryption algorithm on that (valid) ciphertext, like in an usual chosen-ciphertext attack: when there is a decryption failure, this will provide additional information. As a result, the number of decryption failures required to make the attack successful is much lower than in [11], which makes it possible to fully implement the attack in practice and check its efficiency. For instance, for the initial NTRU-1998 parameter sets, a decryption query on a single decryption failure is enough to recover the private key. For more recent parameter sets, the number of required decryption failures increases but is at most a few hundreds. The efficiency of our attacks seems to confirm the importance of removing decryption failures in NTRUEncrypt, as was first suggested in [11]: it should be noted that the latest version [10] of NTRUEncrypt modifies the NTRU parameters so that no decryption failure can ever occur. Furthermore, because we query the decryption oracle on random ciphertexts of messages uniformly chosen at random, our attacks can also be interpreted from a security point of view. If one could simulate the NTRU decryption algorithm, one would be able to recover the NTRU secret key in practice.

Road map. The paper is organized as follows. In Section 2, we provide background on NTRUEncrypt, and we introduce the model of our attacks. In Section 3, we study the probability distributions of the coefficients of the polynomials used during decryption, and we analyze the information obtained in the presence of decryption failures. In Section 4, we derive a first chosen-ciphertext attack against the initial instantiation NTRU-1998 of NTRUEncrypt, which can recover the secret key using a single decryption failure. Finally, in Section 5, we present a general chosen-ciphertext attack against all instantiations of NTRU allowing decryption failures. It is perhaps worth noting that our attacks make no use of lattices.

Acknowledgements. Part of this work is supported by the Commission of the European Communities through the IST program under contract IST-2002-507932 ECRYPT.

2 Background

2.1 Definitions and Notation

NTRUEncrypt operations take place in the quotient ring of polynomials $\mathcal{P} = \mathbb{Z}[X]/(X^N - 1)$, where N is an integer. If $f(X)$ is a polynomial in $\mathcal{P}$, for all $k \in [0, N-1]$, f_k denotes the coefficient of X^k and for all $x \in \mathbb{C}$, $f(x)$ represents the evaluation of f at x. The convolution product $h = f * g$ of two polynomials f and g in $\mathcal{P}$ is given by $h_k = \sum_{i+j \equiv k \bmod N} f_i \cdot g_j$. Several different measures of the size of a polynomial will be useful. We define the *norm* of a polynomial f in the usual way, as the square root of the sum of the squares of its coefficients: $\|f\| = \left(\sum_{i=0}^{N-1} f_i^2\right)^{1/2}$. We also define the *"standard deviation"* of a polynomial f as $\sigma(f) = \left(\sum_{i=1}^{N} \left(f_i - \frac{f(1)}{N}\right)^2\right)^{1/2}$. Note that $\|f\| = \sigma(f)$ if the average value $\frac{f(1)}{N}$ of the coefficients of the polynomial is equal to zero.

2.2 The NTRU Encryption Scheme

The NTRU [8] cryptosystem has many possible instantiations. It uses a set of parameters whose values will be given later:

- An integer N. This fundamental parameter in NTRUEncrypt is taken to be prime to prevent attacks due to Gentry [3], and sufficiently large to prevent lattice attacks.
- Two relatively prime integers p and q, or alternatively the polynomial $p = X + 2$ and a prime number q (which does not divide $2^N + 1$), so that the elements p and q generate prime ideals in $\mathcal{P}$. Standard practice is to take q to be close to a power of 2 between $N/2$ and N, and p to be either the integer 2, 3 or the polynomial $2 + X$ [1,2].
- Four subsets $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_r, \mathcal{L}_m$ of $\mathcal{P}$ used for key generation and encryption. The polynomials in all these subsets have very small coefficients and may further be sparse.
- A bijection ψ between $\mathcal{L}_m \bmod p$ and $\mathcal{L}_m$. The set of plaintexts is $\mathcal{L}_m$.

The key generation, encryption and decryption primitives are as follows:

Key generation.

1: Choose $f \in \mathcal{L}_f$ and $g \in \mathcal{L}_g$ uniformly at random such that f is invertible in $\mathcal{P}$ modulo q and modulo p.
2: Set $F_q = f^{-1} \bmod q$ and $F_p = f^{-1} \bmod p$.
3: The private key is (f, F_p).
4: The public key is $H = p \cdot g * F_q \bmod q$.

Note that in NTRUencrypt, the polynomial g is not necessary for decryption, and therefore is not included in the private key. However, g could easily be deduced from f thanks to $H * f = p \cdot g \bmod q$.

Encryption. The encryption algorithm $\mathfrak{E}$ is probabilistic.

Input: A message $m \in \mathcal{L}_m$, and a public key H.
Output: A ciphertext $e \in \mathfrak{E}(m)$.
1: Select $r \in \mathcal{L}_r$ uniformly at random.
2: **return** $e = r * H + m \bmod q$.

To achieve strong security notions, NTRU implementations additionally use paddings: the message m is preprocessed and r might depend on m and hash functions. In this paper, for any $m \in \mathcal{L}_m$ we denote by $\mathfrak{E}(m)$ the set of all possible ciphertexts of the plaintext m, and by $\mathfrak{E}(\mathcal{L}_m)$ their union over all plaintexts m. These sets take possible paddings into account, hence $\mathfrak{E}(\mathcal{L}_m)$ is also the set of all validly generated ciphertexts.

Decryption. The decryption algorithm $\mathfrak{D}$ provided by NTRU is very efficient, but may not work correctly on all inputs, depending on the parameter set.

Input: A ciphertext $e \in \mathfrak{E}(\mathcal{L}_m)$ and a private key (f, F_p).
Output: A plaintext $\mathfrak{D}(e) = m \in \mathcal{L}_m$.
1: Compute $a \bmod q = e * f \bmod q$.
2: Using a centering procedure, try to recover the integer polynomial $a = p \cdot r * g + f * m \in \mathcal{P}$ from $a \bmod q$.
3: Compute $m \bmod p = a * F_p \bmod p$.
4: **return** The plaintext $m = \psi(m \bmod p)$.

Note that the operations performed by the decryption algorithm could in theory be applied to any polynomial in $\mathcal{P} \bmod q$, not only ciphertexts in $\mathfrak{E}(\mathcal{L}_m)$, and this would still output a polynomial in $\mathcal{L}_m$. The chosen-ciphertext attacks of [13] relied on this extra-functionality: they used invalid ciphertexts which do not belong to $\mathfrak{E}(\mathcal{L}_m)$.

For all the NTRU parameter sets proposed until 2005 [10], the centering procedure used in Step 2 could fail to recover a. Thus, there may exist valid ciphertexts $e \in \mathfrak{E}(m)$ whose decryption $\mathfrak{D}(e)$ is not equal to m. Such events have been called *decryption failures* in [11]. Note that the decryption algorithm could also perform additional checks: for instance, it can extract the random polynomial r used by $\mathfrak{E}$ from the formula $r * H = (e - \mathfrak{D}(e)) \bmod q$, thanks to the notion of pseudo-inverse (see [16]). Then, it can check whether r is really in $\mathcal{L}_r$, or whether the potential paddings requirements are met. In particular, we have a plaintext-checking oracle: given a ciphertext and a plaintext, we can check whether the ciphertext corresponds to the plaintext.

2.3 Instantiations of NTRU

One difficulty with analyzing NTRU is the significant number of variants. Different choices of parameters can completely transform the security of NTRU: an attack against a particular instantiation of NTRU may not work against other instantiations. In this section, we recall the three main instantiations of NTRU which have been proposed in the past eight years.

NTRU-1998. In the initial instantiation of NTRU [8], p is equal to 3 and q is a power of 2 (for example $q = 128$). The subset $\mathcal{L}_m$ is the set of ternary polynomials with coefficients in $\{-1, 0, 1\}$, and the bijection ψ between $\mathcal{L}_m \bmod p$ and $\mathcal{L}_m$ is defined by selecting the representative $-1, 0$ or 1 of each coefficient $\bmod 3$. The set $\mathcal{L}_f$ is defined as $\mathcal{T}(d_f, d_f - 1)$ where $\mathcal{T}(i, j)$ is the subset of ternary polynomials containing exactly i times 1 and j times -1. Finally, $\mathcal{L}_g = \mathcal{T}(d_g, d_g)$ and $\mathcal{L}_r = \mathcal{T}(d_r, d_r)$. Naturally, one drawback with this instantiation is the conversion between binary messages and their ternary representation in $\mathcal{L}_m$. As an example, the parameters $N = 263$, $q = 128$, $d_f = 50$, $d_g = 24$ and $d_r = 16$ were recommended for high security. Each parameter set in [8] leads to a decryption failure every 2^{15} encryptions of random messages, experimentally.

NTRU-2001. In the standards [1,2], a new instantiation is proposed, where p is the polynomial $X + 2$ and q is a prime number. The subset $\mathcal{L}_m$ is the set of binary polynomials with coefficients in $\{0, 1\}$, and $\mathcal{L}_f$ is the subset of polynomials of the form $1 + p * F$ with $F \in \mathcal{B}(d_F)$, where $\mathcal{B}(d_F)$ denotes the set of binary polynomials with exactly d_F coefficients equal to 1. The other subsets are $\mathcal{L}_g = \mathcal{B}(d_g)$ and $\mathcal{L}_r = \mathcal{B}(d_r)$. The bijection ψ between a plaintext m and its representative modulo $X + 2$ (which is the evaluation at $X = -2$) is non-trivial. Mathematically, the function ψ computes the binary decomposition $\sum_{i=0}^{N-1} \nu_i 2^i$ of $m(-2)$, and identifies it with the polynomial $\sum_{i=0}^{N-1} \nu_i X^i \in \mathcal{L}_m$. More details for an efficient implementation are given in [9]. The main advantage of having the private key f of the form $1 + p * F$ is that the inverse F_p modulo p is equal to 1, so the final multiplication by F_p in the decryption process disappears. The average number of encryptions of random messages leading to a decryption failure ranges from 2^{12} to 2^{25}, depending on the parameter set [1,2]. (see [11] for more information).

NTRU-2005. In the last standard [10], the polynomial $p = X+2$ disappears and is replaced by the integer $p = 2$. Furthermore, the use of product-form polynomials (introduced in [9]) is recommended as a replacement of binary polynomials: so f has the form $1 + p \cdot F$ with $F \in \mathcal{X}(d_f)$, which means that $F = f_1 * f_2 + f_3$ with each $f_1, f_2, f_3 \in \mathcal{B}(d_f)$. The other subsets are $\mathcal{L}_g = \mathcal{B}(N/2)$, $\mathcal{L}_r = \mathcal{X}(d_r)$ and $\mathcal{L}_m$ is the set of binary polynomials. Generally, d_F and d_r are equal and are very small (*e.g.* between $N/25$ and $N/20$). More importantly, since it had been discovered [11] that decryption failures could be a threat, the prime number q has been multiplied by a factor of at least 2 so that no decryption failure can ever happen: one drawback is that the resistance to lattice attacks is weakened. But it is interesting to analyze what would happen if q had the same size as in previous instantiations. In this case, the problem of finding the private key from the public key seems as hard as in previous instantiations. But the proportion of decryption failures would also be the same as in the previous instantiations. In this paper, we want to analyze attacks based on decryption failures, so when we refer to NTRU-2005, we actually mean a modified version with a smaller q.

2.4 The attack model, and comparison with previous attacks

In a chosen-ciphertext attack, a decryption oracle is given to an attacker. As a precomputation, the attacker can submit as many ciphertexts as he wants to the oracle. Then using the collected information, he must either recover a private key or be able to decrypt a challenge ciphertext. For NTRU, the notion of decryption oracle is ambiguous because of decryption failures. Here, like [11], by decryption oracle, we do not mean an ideal algorithm which would extract m from a ciphertext in $\mathfrak{E}(m)$ without any failure, but the decryption algorithm $\mathfrak{D}$ provided by NTRU. In the following, we only consider key-recovery chosen-ciphertext attacks.

The majority of previous chosen-ciphertext attacks against NTRU work by running the algorithm $\mathfrak{D}$ on special polynomials in $\mathcal{P} \bmod q$, which are generally not valid ciphertexts in $\mathfrak{E}(\mathcal{L}_m)$. Following our terminology, these attacks do not use decryption failures. For example, the article of Jaulmes and Joux at CRYPTO'00 [13] presents two chosen-ciphertext attacks on NTRU-1998. By sending roughly ten special polynomials to the oracle, an attacker recovers the product of the key F_p and a low hamming-weight polynomial. After an exhaustive search which takes a couple of minutes for the highest security parameters, the attacker recovers F_p and deduces the private key f. Note that the bijection between F_p and f only exists in the NTRU-1998 instantiation.

The second attack of [13] queries the decryption oracle N times on very close inputs. Again, the input polynomials are in general not validly generated ciphertexts in $\mathfrak{E}(\mathcal{L}_m)$. If f is binary or ternary, the output of the decryption oracle then discloses the value and the position of many coefficients of f. Thus, with less than a thousand calls to the decryption oracle, the private key f is fully recovered. This attack can be rewritten and remains valid for NTRU-2001, but fails on NTRU-2005, when f is in product form.

Other papers, like Han *et al.*'s paper [6], present chosen-ciphertext attacks with the assumption that the user has power on m and r, which is not compatible with the strongest paddings.

The first paper to introduce and use decryption failures in a chosen-ciphertext attack is Howgrave-Graham *et al.*'s paper [11] at CRYPTO'03, where the authors present (among others) a key-recovery chosen-ciphertext attack against NTRU-2001 working with any padding scheme. The oracle is weak: it only accepts valid ciphertexts in $\mathfrak{E}(\mathcal{L}_m)$ and only indicates whether or not there is a decryption failure. The authors of [11] claim that if they are given a million decryption failures, they can recover the polynomial $X^N f(X) f(\frac{1}{X})$, and then recover the private key thanks to an algorithm of Gentry and Szydlo [5], which was introduced to break a former version of NTRU signatures. The main advantage of this attack is that since all messages are validly generated, it is compatible with any padding, including the very restrictive ones. However, the number 1,000,000 is only a heuristic estimate, and the algorithm of Gentry and Szydlo [5] recovering f from $X^N f(X) f(\frac{1}{X})$ is proved polynomial time, but has to our knowledge not yet been fully implemented in practice.

In this paper, we also use decryption failures. Our attack model is intermediate between the restrictive chosen-ciphertext attack of Jaulmes and Joux [13], and the realistic model in Howgrave-Graham *et al.*'s attack [11]. The oracle is only queried during the search for decryption failures, which is performed by Algorithm 1. With this description, we clearly see that the decryption oracle is

Algorithm 1 Find a random decryption failure

Input: A NTRU parameter set, a public key H and the decryption oracle $\mathfrak{D}$.
Output: A decryption failure as (m, r, m') where $\mathfrak{D}(m + r * H) = m' \neq m$.

1: **repeat**
2: Generate a random message $m \in \mathcal{L}_m$ and encrypt it with $\mathfrak{E}$ to obtain a valid ciphertext e.
3: Remember (or recover) the random polynomial r used by $\mathfrak{E}$.
4: Submit the ciphertext e to the decryption oracle $\mathfrak{D}$.
5: **until** there is a decryption failure ($m' = \mathfrak{D}(e) \neq m$)
6: **return** the triplet (m, r, m').

only used on validly generated ciphertexts. Furthermore, these ciphertexts are not even chosen by the attacker, but are randomly generated. For these reasons, the attacker is less powerful than in Jaulmes and Joux' attack [13]. However, the attacker has access to the output m', which gives more information than in Howgrave-Graham *et al.*'s attack [11]. We will see in the next sections the number of decryption failures which is necessary to recover the private key.

3 Analysis of decryption failures

3.1 The decryption process

In Section 2.2, we only gave a sketch of the decryption primitive. In this section, we give a detailed implementation (see Algorithm 2), in order to explain decryption failures. In the first step of the decryption algorithm $\mathfrak{D}$, when calculating $a \bmod q = f * e \bmod q$, one actually computes

$$a \bmod q = f * e = f * (r * h + m) = p \cdot r * g + f * m \bmod q$$

The polynomials r, g, f and m have very small coefficients, so heuristically the coefficients of the integer polynomial $a = p \cdot r * g + f * m$ satisfy:

$$\forall i \in [0..N-1],\ a_i \in \left[\frac{a(1)}{N} - \frac{q}{2}; \frac{a(1)}{N} + \frac{q}{2}\right[. \tag{1}$$

Algorithm 2 Decryption oracle $\mathfrak{D}$

Input: A validly generated ciphertext e and the instantiation of NTRU (and a hidden private key (p, f, F_p)).
Output: The decryption of e (which might be incorrect).
1: compute $a \bmod q = f * e \bmod q$.
2: **if** instantiation=NTRU-1998 **then**
3: Select the representative $a_{\text{est}} \in \mathcal{P}$ of $a \bmod q$ which has all its coefficients in $[-\frac{q}{2}; \frac{q}{2}[$.
4: **else**
5: Compute $r1 = r(1)$,$f1 = f(1)$, and $g1 = g(1)$ using the definition of $\mathcal{L}_r$,$\mathcal{L}_f$ and $\mathcal{L}_g$.
6: Compute $m(1) \bmod q = e(1) - r1 * H(1) \bmod q$,
7: Choose $m1$ in $[\frac{N}{2} - \frac{q}{2}; \frac{N}{2} + \frac{q}{2}[$ so that $m1 \equiv m(1) \bmod q$.
8: Compute $a1 = p \cdot r1 \cdot g1 + f1 \cdot m1$.
9: Select the representative $a_{\text{est}} \in \mathcal{P}$ of $a \bmod q$ which has all its coefficients in $[\frac{a1}{N} - \frac{q}{2}; \frac{a1}{N} + \frac{q}{2}[$
10: **end if**
11: Compute $m' = a_{\text{est}} * F_p \bmod p$.
12: **return** $\psi(m')$.

Note that all the parameter sets of NTRU given in Section 2.3 make it possible to compute $\frac{a(1)}{N}$ without knowing r or m. If Condition (1) holds, a may be recovered exactly from $a \bmod q$ in the ring $\mathcal{P}$ (using Step 9 of Algorithm 2). In the NTRU-1998 instantiation, the mean value $\frac{a(1)}{N}$ of the coefficients of a is equal to $\frac{m(1)}{N}$, and is therefore between -1 and 1. For this reason, it is equivalent to choose every coefficient of a in $[-\frac{q}{2}; \frac{q}{2}[$ at Step 3. Finally $a \bmod p$ is equal to $f * m \bmod p$ and the multiplication by F_p at Step 11 recovers $m \bmod p$, and therefore the plaintext m.

Decryption only works if the condition (1) is fulfilled. Unfortunately, this condition does not always hold: depending on the choice of $\mathcal{L}_m$ and $\mathcal{L}_r$, it may happen for some rare m and r, that some coefficients of a lie outside the centering range. In this case, the output $m' = \mathfrak{D}(e)$ will (almost always) differ from the original plaintext m. These events are the so-called *decryption failures* [11], which will be reused in this paper to construct key-recovery attacks on NTRUEncrypt.

We now analyze the probability distribution of the coefficients of a in the particular case of a decryption failure. In order to simplify Condition (1), it is possible to translate the polynomials so that the average value of their coefficients is always zero. We say that a polynomial a is zero-centered if $a(1) = 0$, where $a(1)$ is the evaluation of the polynomial at 1, that is, the sum of the coefficients of a. Given any polynomial in $\mathcal{P}$ or $\mathcal{R}$, we can recenter this polynomial by subtracting an appropriate multiple of the polynomial $(1 + X + \cdots + X^{N-1})$, as shown in the following elementary lemma:

Lemma 1. *The following function is an algebra homomorphism:*

$$\begin{aligned}\mathbb{R}[X]/\left(X^N-1\right) &\to \mathbb{R}[X]/\left(X^N-1\right)\\ A \to \check{A} &= A - \frac{A(1)}{N}(1+X+\cdots+X^{N-1})\end{aligned}$$

Then Condition (1) can be rewritten as: there is a decryption failure if and only if the polynomial $\check{a} = p \cdot \check{r} * \check{g} + \check{f} * \check{m}$ satisfy:

$$\exists i \in [0..N-1],\ |\check{a}_i| > \frac{q}{2}. \tag{2}$$

3.2 Probability assumptions

In the following, we will often need to assume that certain objects are random. More precisely, for any deterministic function φ from $\mathcal{L}_m \times \mathcal{L}_r$ (*e.g.* the encryption function $(m,r) \to m+r*H$ or the function $(m,r) \to p \cdot r*g+f*m$ implicitly used in the decryption process), we say that a polynomial $z = \varphi(m,r)$ is randomly chosen in the image of φ if m and r are uniformly and independently chosen in the finite subsets $\mathcal{L}_m$ and $\mathcal{L}_r$. Here, we focus on the particular case of the centered polynomials $\check{a}$, which are computed from m and r with the deterministic formula $\check{a} = p \cdot \check{r} * \check{g} + \check{f} * \check{m}$. In order to analyze the distribution of their coefficients, we need to make two simplifying assumptions:

Assumption 1. *If $z \in \mathcal{P}$ is a binary or ternary polynomial uniformly chosen at random, then the coefficients of the zero-centered polynomial $\check{z}$ are all independent.*

Assumption 2. *If a and b are randomly and independently drawn from finite sets of zero-centered polynomials and if their coefficients are all independent, then the coefficients of the product $c = a * b$ are all independent.*

These assumptions are rather strong. However, one can hope that it is not that far from the reality, due to the following: the average value of the coefficients of a zero-centered polynomial is constant and equal to zero, so when a coefficient is bigger than expected, the others will tend to be smaller. For this reason, there is a small anti-correlation between different coefficients: the paper [17] shows a correlation $\mathrm{Corr}(c_i, c_j) = -\frac{1}{N-1}$ if i and j are distinct indexes. There is a small inaccuracy due to this anti-correlation, but the effect is very small when N grows. Furthermore, we will see that experimentally, if one coefficient has not the expected size, then the others behave correctly. Thus the effect of the anti-correlation is in fact very limited.

3.3 Shape of decryption failures

In the decryption algorithm $\mathfrak{D}$, since r and m are randomly chosen independently of the keys f and g, the coefficients of the polynomial $\check{a}$ are assumed to be independent by Assumptions 1 and 2. As we saw in Section 3.1, a decryption failure only occurs if Condition (2) holds. In this case, at least one coefficient of the polynomial $\check{a} = p \cdot \check{g}\check{r} + \check{f}\check{m}$ is outside the range $[-\frac{q}{2}; \frac{q}{2}[$. Using these assumptions, we deduce two heuristics:

Heuristic 1. *In case of a decryption failure, then with extremely high probability, there is exactly one coefficient $\check{a}_k$ of $\check{a}$ which is outside $[-\frac{q}{2}; \frac{q}{2}[$*

Explanation. We denote by p_e the probability to choose (r, m) leading to a decryption failure. Because of the independence assumption, the probability for one coefficient of $\check{a}$ to be outside the range $[-\frac{q}{2}; \frac{q}{2}[$ is $p_a = 1 - (1 - p_e)^{\frac{1}{N}} \approx \frac{1}{N} p_e$. The probability that exactly one coefficient of $\check{a}$ is too big, is $p_{\text{one}} = \binom{N}{1} \cdot p_a(1 - p_a)^{N-1} \approx p_e(1 - \frac{N-1}{N} p_e)$. Thus in case of a decryption error, the probability that only one coefficient of $\check{a}$ is too big is $p_{\text{one}}/p_e \approx (1 - \frac{N-1}{N} p_e)$. Since the probability of decryption failure p_e is always very small, the last probability is almost equal to 1. □

When a decryption failure occurs, all the coefficients except one are in the correct interval. The second heuristic guesses the value of this *overflowing coefficient.*

Heuristic 2. *In case of a decryption failure, the overflowing coefficient of $\check{a}_k$ defined by Condition (2) is very close to $\pm\frac{q}{2}$. More precisely, for standard NTRU parameters, we expect that $\check{a}_k = \pm\left(\frac{q}{2} + \varepsilon\right)$ where $\varepsilon \leq 5$.*

Explanation. The distribution of a coefficient $\check{a}_k$ should be a discrete hypergeometric distribution whose mean is 0, and whose standard deviation is smaller than $\frac{q}{4}$. If so, it would decrease much faster than a geometric distribution for the rare values greater than $\frac{q}{2}$. Then the expectation of the value of the overflowing coefficient of $\check{a}$ would be very close to $\frac{q}{2}$. □

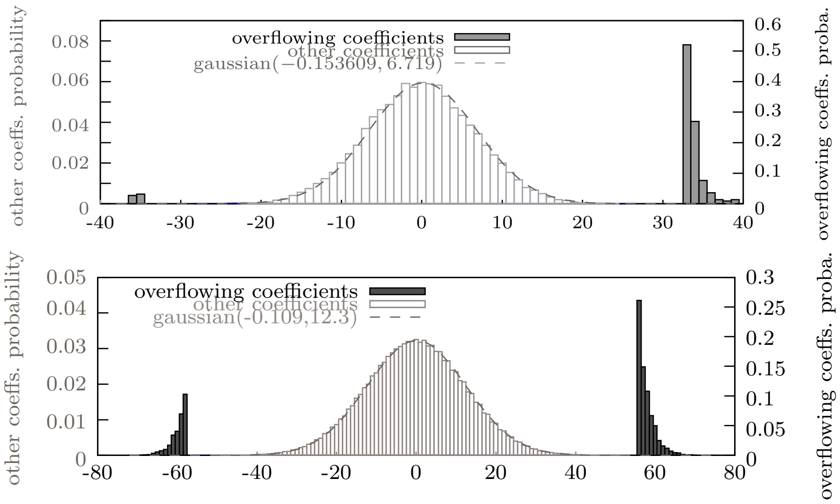
3.4 Experiments

To check the validity of our heuristics, we performed experiments on the main NTRU instantiations: NTRU-1998, NTRU-2001 and the slightly modified version of NTRU-2005 seen in Section 2.3. We obtained approximately one decryption failure every 25,000 messages, and we collected about 4,000 decryption failures for each parameter set. For every decryption failure, we obtained exactly one overflowing coefficient in $\check{a}$, and in more than 90% of the cases, the absolute value of this coefficient was lower than $\frac{q}{2} + 5$. So the two heuristics seem to be verified in practice.

However, in our experiments, the distribution of overflowing coefficients in case of a decryption failure does not decrease as fast as was announced in the explanation of Heuristic 2 (see Figure 1). Heuristic 2 is nevertheless true for every parameter set tested, but this is only an experimental fact.

The two graphs in Figure 1 represent the experimental distribution of the coefficients of a random polynomial $\check{a} = p \cdot \check{r} * \check{g} + \check{f} * \check{m}$ satisfying Condition (2). The black bars represent the distribution of the overflowing coefficient (whose

Fig. 1. Experimental densities:



absolute value is greater than $\frac{q}{2}$), and the grey boxes represent the distribution of the other coefficients. The two curves are based on the NTRU-2005 instantiation, the first one with $p = 2, q = 67, N = 127$ and binary polynomials, and the second one with $p = 2, q = 67, N = 127$ and product-form polynomials. When the polynomial $\breve{a}$ contains an overflowing coefficient, there exists a rotation of $\breve{m}$ (resp. $\breve{r}$) which is either positively or negatively correlated with the key $\breve{f}$ (resp. $p \cdot \breve{g}$), depending on whether the overflowing coefficient is positive or negative (see Section 5). In these two particular examples, there are more positive correlations, but this is not the general case.

4 A new chosen-ciphertext attack against NTRU-1998

We will now describe chosen-ciphertext attacks on NTRU which only require few decryption failures to recover the private key. In this section, we consider the special case of NTRU-1998. The first attack uses the fact that a single decryption failure is enough to recover F_p up to a shift. It turns out that in NTRU-1998, the choice of $p = 3$ and f a ternary polynomial makes it possible to recover f from $f \bmod p$ or $F_p \bmod p$. Thus, one can recover a circular shift of the private key f in the quotient ring $\mathcal{P}$, which is enough to decrypt. The attack is summarized in Algorithm 3.

4.1 Description of the attack

The parameters sets of [8] give rise to a decryption failure every 25,000 encryptions, independently of the security level. Thus, after say 50,000 calls to the

Algorithm 3 A key-recovery chosen-ciphertext attack on NTRU-1998

Input: A public key H of NTRU-1998.
Output: A private key (f, F_p).
1: Find a decryption failure (m, r, m') using Algorithm 1.
2: compute $F'_p \leftarrow (m' - m) \bmod p$ **where** $p = 3$.
3: $f' \leftarrow F_p'^{-1} \bmod p$ (chose the coefficients of f' in $\{-1, 0, 1\}$.
4: **if** $g' = (f' * H * p^{-1}) \bmod q \notin \mathcal{L}_g$ **restart;.**
5: **return** (f', F'_p).

decryption oracle, we may reasonably assume that we were able to find a message $m \in \mathcal{L}_m$ and a blinding polynomial $r \in \mathcal{L}_r$ leading to a decryption failure, and we know the polynomial $m' \neq m$ returned by the decryption oracle. Then if Heuristic 1 is satisfied, there is exactly one coefficient a_k of $a = prg + fm$ which is not in the centering range. Thus, there exists an integer $\alpha \in \mathbb{Z}$ such that $a - \alpha q X^k$ satisfies Condition (1), and it is precisely this polynomial which is computed instead of a during the decryption algorithm. In the last step of the decryption, $a - \alpha q \cdot X^k$ is taken modulo p and multiplied by F_p, so the incorrect plaintext returned is:

$$m' = F_p * (fm - \alpha q \cdot X^k) \bmod p = m - (\alpha q \bmod p) \cdot X^k * F_p.$$

Since there was a decryption failure and since $p = 3$, we are sure that $\alpha' = \alpha q \bmod p$ is equal to ± 1. Therefore, by considering the difference $m - m'$, we recover a rotation (up to a sign) $F'_p = \alpha' X^k * F_p$ of the polynomial F_p from the private key. It is then easy to invert $F'_p \bmod p$ in order to find the second part of the key $f' = \alpha' X^{N-k} * f$. As we saw at the beginning of the section, the computed polynomials (f', F'_p) form a secret key equivalent to (f, F_p), so it is not necessary to find the value of k.

4.2 Experiments

We have implemented the attack described by Algorithm 3. In practice, the first decryption failure is found before 40,000 calls to the oracle $\mathfrak{D}$. As seen previously, the probability for (f', F'_p) computed in steps 2 and 3 to be a valid private key is higher than $1 - p_e \approx \frac{24999}{25000}$. In this case, the polynomial g' computed in step 4 is a rotation of g, and the algorithm ends. We chose 4 keys at random in each parameter set of [8], and we collected 50 decryption failures per key. Instead of stopping the execution once a circular shift of the key had been found, we ran the attack on all the decryption failures obtained on the NTRU-1998 instantiation. Each one disclosed a rotation (up to a sign) of the private key, which is an equivalent private key.

5 A general attack against all NTRU instantiations

The attack on NTRU-1998 seen in the last section cannot be applied to more recent instantiations of NTRU. Indeed, in the original implementation of NTRU, the choice of $p = 3$ and a ternary key f made it possible to recover entirely f from F_p . But in recent optimizations, the choice of $f = 1 + p \cdot F$ implies that F_p is always equal to 1, and does not leak any information about f. However, it is still possible to construct a key-recovery algorithm, using this time a few hundreds of decryption failures. Our attack builds on [11], where it was noticed that each decryption failure gave rise to an approximation of an (unknown) rotation of f and g. Here, we further notice that the full output of the decryption oracle enables us to find out which rotation it is. In other words, the output of the oracle on each decryption failure discloses two polynomials, which have a majority of coefficients in common with the secret keys f and g (without any rotation). Unfortunately, the approximation provided by the decryption oracle does not reveal the exact position of the correct coefficients. In this attack, we therefore average many approximations of f deduced from independent decryption failures, in order to compute a more accurate approximation, until f can be recovered by rounding. The attack will be described with p being an integer. However it works on every instantiation of NTRU, even if p is the polynomial $X + 2$. In this case, the only difference is that we have to replace the multiplication $p \cdot g$ by the convolution product $p * g$. The attack is summarized by Algorithm 4.

Algorithm 4 A general key-recovery chosen-ciphertext attack

Input: A public key H.
Output: A secret key (f, F_p).

$p \leftarrow 0$; $\boldsymbol{S} \leftarrow \boldsymbol{0}$.
estimate $\alpha \leftarrow \frac{2}{2\left(\|\check{f}\|^2 + \|p \cdot \check{g}\|^2\right)}$ using NTRU parameters.
loop
  Find a decryption failure (m, r, m') using Algorithm 1.
  Find the integer k such that $m_k \neq m'_k$.
  $\boldsymbol{V} \leftarrow (m_k, m_{k-1}, \ldots, m_{k-N+1 \bmod N}, r_k, r_{k-1}, \ldots, r_{k-N+1 \bmod N})$.
  $\epsilon \leftarrow \mathrm{sign}(\langle \boldsymbol{V}, \boldsymbol{S} \rangle)$ or $+1$ if $\langle \boldsymbol{V}, \boldsymbol{S} \rangle = 0$.
  $\boldsymbol{S} \leftarrow \boldsymbol{S} + \epsilon \boldsymbol{V}$; $z \leftarrow z + 1$.
  let $f = \sum_{i=0}^{N-1} \mathrm{round}(\frac{1}{\alpha z} \boldsymbol{S}_1 + \frac{f(1)}{N}) X^i$.
  if $f * H * p^{-1} \bmod q \in \mathcal{L}_g$, **then return** $(f, f^{-1} \bmod p)$.
  let $f' = \sum_{i=0}^{N-1} \mathrm{round}(-\frac{1}{\alpha z} \boldsymbol{S}_1 + \frac{f(1)}{N}) X^i$.
  if $f' * H * p^{-1} \bmod q \in \mathcal{L}_g$, **then return** $(f', f'^{-1} \bmod p)$.
end loop.

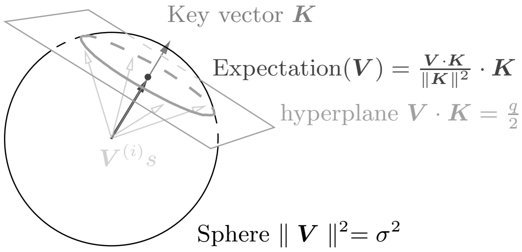
5.1 Description of the attack

Again, we assume that we were able to find $m \in \mathcal{L}_m$ and $r \in \mathcal{L}_r$ such that the ciphertext $m + rH$ is decrypted as $m' \neq m$. This time, m and r are not zero-centered, so it is necessary to use the centering homomorphism in order to use Condition (1). From Heuristic 1, we know that there exists $\epsilon = \pm 1$ and $k \in [0, N-1]$ such that $p \cdot \check{r} * \check{g} + \check{m} * \check{f} - \epsilon q \cdot X^k$ has all its coefficients in $[-\frac{q}{2}; \frac{q}{2}[$. Thus, the output of the decryption algorithm is $m' = m + X^k \bmod p$, and k is therefore the only index where the coefficients m and m' differ. Hence, the value of k is disclosed and the k^{th} coefficient is the overflowing coefficient, so:

$$\left|\sum_{j=0}^{N-1} \check{m}_{k-j \bmod N} \check{f}_j + \sum_{j=0}^{N-1} \check{r}_{k-j \bmod N} p \check{g}_j\right| \geq \frac{q}{2}$$

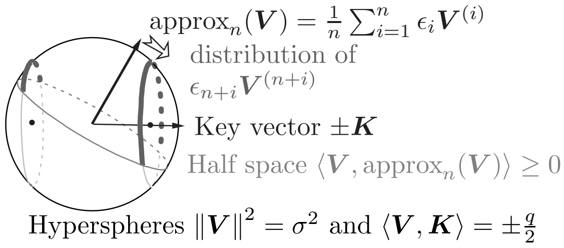
If we call $\boldsymbol{V}$ the vector $(\check{m}_{(k-0) \bmod N}, \ldots, \check{m}_{(k-N+1) \bmod N}, \check{r}_{(k-0) \bmod N}, \ldots, \check{r}_{(k-N+1) \bmod N})$ and $\boldsymbol{K}$ the vector $(\check{f}_0, \ldots, \check{f}_{N-1}, p \cdot \check{g}_0, \ldots, p \cdot \check{g}_{N-1})$ representing the private key, the previous inequality can be rewritten as $|\langle \boldsymbol{V}, \boldsymbol{K} \rangle| \geq \frac{q}{2}$. Thus, either $\boldsymbol{V}$ is very correlated with $\boldsymbol{K}$ or it is strongly anti-correlated. And $\boldsymbol{K}$ is in one-to-one correspondence with the private key. Note that since the sum of the coefficients of $\boldsymbol{V}$ is equal to zero, the squared norm $\|\boldsymbol{V}\|^2$ is the sum of the variance of m and r. It is a constant σ^2 depending only on the NTRU parameter set. Likewise, $\|\boldsymbol{K}\|^2$ is the sum of the variances of the keys f and g, and it depends only on the parameter set. For instance, in NTRU-2005 with binary polynomials, the private key f is equal to $1 + 2F$ where F is binary and contains exactly d_F ones. Therefore $\|f\|^2 = \sum_{j=0}^{N-1} f_j^2 = 1 + 4d_F$. After centering the polynomial, $\left\|\check{f}\right\|^2 = \|f\|^2 - \frac{1}{N} f^2(1)$ where $f(1) = 1 + 2d_F$, so $\left\|\check{f}\right\|^2 = 4 \cdot \frac{d_F(N-d_F-1)}{N}$. Using the same kind of arguments for $\|p \cdot \check{g}\|^2$, we show that $\|\boldsymbol{K}\|^2 = \frac{4}{N} \cdot (d_F(N - d_F - 1) + 2d_g(N - d_g))$ in NTRU-2005.

Fig. 2. Simplified case



Simplified case. In the simplified case, only positive correlations occur, and Heuristic 2 suggests that the inequality $|\langle \boldsymbol{V}, \boldsymbol{K} \rangle| \geq \frac{q}{2}$ is in fact almost an equality. Then, as shown in Figure 2, such a vector $\boldsymbol{V}$ is located on the hypersphere at the intersection of the sphere of equation $\|\boldsymbol{V}\|^2 = \sigma^2$ and the hyperplane of equation $|\langle \boldsymbol{V}, \boldsymbol{K} \rangle| = \frac{q}{2}$. If we gather many independent decryption failures, we will obtain many $\boldsymbol{V}^{(i)}$'s in this hypersphere. Their expectation is the center of the hypersphere, which is the multiple $\frac{q}{2||K||^2} \cdot \boldsymbol{K}$ of the private key vector. Therefore if we consider n vectors $\boldsymbol{V}^{(i)}$s coming from independent decryption failures, then their mean value $\frac{1}{n} \sum_{i=1}^{n} \boldsymbol{V}^{(i)}$ shall converge to $\frac{q}{2||K||^2} \cdot \boldsymbol{K}$ when n grows. In practice, $\frac{q}{2||K||^2}$ seems to be greater than $\frac{1}{16}$ in every instantiation of NTRU containing decryption failures. For this reason, if n is of the order of N, we expect to have enough accuracy to recover the full key $\boldsymbol{K}$.

Fig. 3. Real case



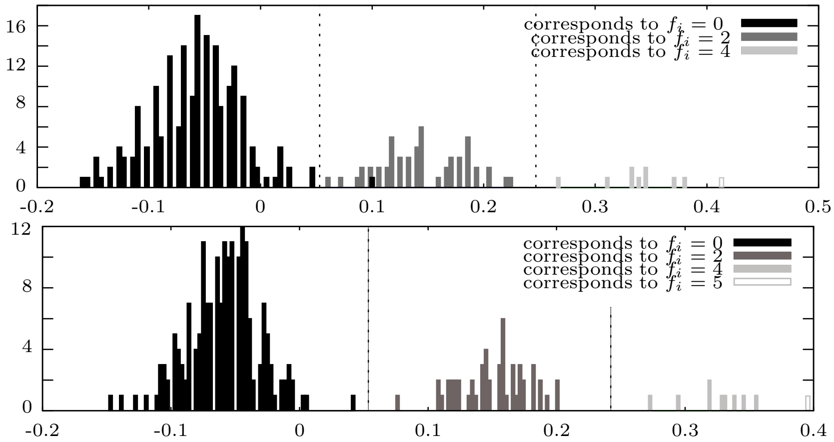
Real case. In reality, there may be both positive and negative correlations between $\boldsymbol{V}$ and $\boldsymbol{K}$. Therefore, $\boldsymbol{V}$ is located on the union of two opposite hyperspheres $\|\boldsymbol{V}\|^2 = \sigma^2 \cap \langle \boldsymbol{V}, \boldsymbol{K}\rangle = \pm\frac{q}{2}$ (see Figure 3). Unfortunately, the proportion of positive and negative correlations (which depends on the fractional part of $\frac{a(1)}{N}$) may be equal to $\frac{1}{2}$ in the worst case. In this case, the expectation of $\boldsymbol{V}$ is zero. We must be able to decide whether the correlation between $\boldsymbol{V}$ and $\boldsymbol{K}$ is positive or not. The best test would be to compute directly the dot product $\langle \boldsymbol{V}, \boldsymbol{K}\rangle$, but we do not know $\boldsymbol{K}$. Therefore, in our algorithm, we try to guess for each $\boldsymbol{V}^{(i)}$, a sign $\epsilon_i = \pm 1$ such that $\langle \epsilon_i \boldsymbol{V}^{(i)}, \boldsymbol{K}\rangle \geq \frac{q}{2}$ for all i, or $\langle \epsilon_i \boldsymbol{V}^{(i)}, \boldsymbol{K}\rangle \leq -\frac{q}{2}$ for all i. In order to do that, we arbitrarily set $\epsilon_1 = 1$, and recursively set $\epsilon_{n+1} = \mathrm{sign}(\langle \boldsymbol{V}^{(n+1)}, \sum_{i=1}^{n} \epsilon_i {}^i\boldsymbol{V}\rangle)$, hoping that $\sum_{i=1}^{n} \epsilon_i \boldsymbol{V}^{(i)}$ is not orthogonal to $\boldsymbol{K}$ (as shown in Figure 3). Then, as suggested on the figure, the sum $\sum_{i=1}^{n} \epsilon_i \boldsymbol{V}^{(i)}$ will slowly take the direction of $\boldsymbol{K}$ or $-\boldsymbol{K}$, and like in the simple case, the average vector $\frac{1}{n}\sum_{i=1}^{n} \epsilon_i \boldsymbol{V}^{(i)}$ will converge to $\pm\frac{q}{2\|K\|^2} \cdot \boldsymbol{K}$.

5.2 Experiments

We have implemented the attack described by Algorithm 4. Both the encryption, decryption algorithms and the attack were implemented using the NTL library [15] without any running-time optimization. Indeed, since the attack only consists of adding a few hundreds of vectors, its running time is negligible compared to the time of collecting the required number of decryption failures. We refer to [8,10,9] for the actual running time of an efficient implementation of the encryption and the decryption algorithm.

As shown in Table 1, the number of decryption failures needed in order to fully recover the private key (even for highest parameters) is a few hundreds. Recall that there is a decryption failure every 2^{15} encryptions: this means that the total number of calls to the decryption oracle is about 2^{23}, which takes less than two days with an unoptimized version of the encryption and decryption algorithms. As shown in Figure 4, the main idea of the algorithm is to build an approximation of the private key vector. The more decryption failures we use, the more accurate the approximation. Since the secret key has integer coefficients, the approximation eventually reaches a precision level where a simple rounding procedure is enough to fully recover the key.

Fig. 4. Coefficients of the estimation of the key in the algorithm:



These two graphs represent the distribution of the coefficients of the estimation of the key (that is the vector S/z of Algorithm 4 line 9) for NTRU-2005 with product-form keys with $N = 251$, $q = 113$. The different levels of grey represent the value of the corresponding key coefficient. The attack works only if the different colors are well separated (by the dotted vertical lines). The first graph is obtained after gathering 150 decryption failures, and the second after 250 decryption failures. In the first graph, there is only one black bar which is misplaced (in the $f_i = 2$ area, so after 150 decryption failures, we have recovered all bits of f with exactly 1 error. In the second graph, the colors are well separated, so the private key is fully recovered.

Table 1. Experiments

type NTRU	N	p	q	d_F	number of decryption failures used	number of digits of f recovered
NTRU-1998	167	3	128	32	50 100	N-3 N (all)
NTRU-2001	251	X+2	127	40	80 140	N-1 N (all)
NTRU-2005 binary (using a smaller q)	251	2	127	64	80 130	N-2 N (all)
NTRU-2005 product form (using a smaller q)	251	2	113	9	100 150 250	N-3 N-1 N (all)

5.3 Potential Improvements

Here, we discuss potential improvements which might further lower the number of required decryption failures.

Exhaustive search. The number of errors in Table 1 between the private key and the estimation is very small during the last half of the algorithm. If we assume that there are less than three errors of at most one unit in our estimation, then the number of possible keys is bounded by $8\binom{N}{2} + 4\binom{N}{2} + 2N$. Even with $N = 251$, it is possible to perform this exhaustive search in practice. In Table 1, the number of decryption failures needed to recover the private key up to 3 errors is half of the number required to fully recover the key. Hence, exhaustive search would divide the number of calls to the decryption oracle by a factor 2.

Lattice attack. Instead of performing an exhaustive search, we may use a lattice reduction algorithm. Once we get a sufficiently accurate approximation of the vector $\boldsymbol{K} = (f, p \cdot g)$, the distance between this approximation of $\boldsymbol{K}$ and the lattice generated by the NTRU public basis should be extremely small compared to its shortest vector. In this case, we may hope that lattice reduction algorithms can recover the whole private key from this approximation in practice.

References

1. Consortium for Efficient Embedded Security. Efficient embedded security standards #1: Implementation aspects of NTRU and NSS, 2001.
2. Consortium for Efficient Embedded Security. Efficient embedded security standards #1: Implementation aspects of NTRUEncrypt and NTRUSign, 2002.
3. C. Gentry. Key recovery and message attacks on NTRU-composite. In *Proc. of Eurocrypt '01*, volume 2045 of *LNCS*. IACR, Springer-Verlag, 2001.
4. C. Gentry, J. Jonsson, J. Stern, and M. Szydlo. Cryptanalysis of the NTRU signature scheme (NSS) from Eurocrypt 2001. In *Proc. of Asiacrypt '01*, volume 2248 of *LNCS*. Springer-Verlag, 2001.
5. C. Gentry and M. Szydlo. Cryptanalysis of the revised NTRU signature scheme. In *Proc. of Eurocrypt '02*, volume 2332 of *LNCS*. Springer-Verlag, 2002.
6. D. Han, J. Hong, J. W. Han, and D. Kwon. Key recovery attacks on NTRU without ciphertext validation routine. *ACISP 2003*, LNCS 2727:pages 274–284, 2003.
7. J. Hoffstein, N. A. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. In *Proc. of CT-RSA*, volume 2612 of *LNCS*. Springer-Verlag, 2003.
8. J. Hoffstein, J. Pipher, and J. Silverman. NTRU: a ring based public key cryptosystem. In *Proc. of ANTS III*, volume 1423 of *LNCS*, pages 267–288. Springer-Verlag, 1998. First presented at the rump session of Crypto '96.
9. J. Hoffstein and J. H. Silverman. Optimizations for NTRU. In *Public-key Cryptography and Computational Number Theory*. DeGruyter, 2000. available at `http://www.ntru.com`.
10. N. Howgrave-Graham, J. H. Silverman, and W. Whyte. Choosing parameter sets for NTRUEncrypt with NAEP and SVES-3.
11. N. A. Howgrave-Graham, P. Q. Nguyen, D. Pointcheval, J. Proos., J. H. Silverman, A. Singer, and W. Whyte. The impact of decryption failures on the security of NTRU encryption. In *Proc. of the 23rd Cryptology Conference (Crypto '03)*, volume 2729 of *LNCS*, pages 226–246. IACR, Springer-Verlag, 2003.
12. IEEE. *P1363.1 Public-Key Cryptographic Techniques Based on Hard Problems over Lattices*, June 2003. IEEE., Available from `http://grouper.ieee.org/groups/1363/lattPK/index.html`.
13. E. Jaulmes and A. Joux. A chosen ciphertext attack on NTRU. In *Proc. of Crypto '00*, volume 1880 of *LNCS*. IACR, Springer-Verlag, 2000.
14. P. Q. Nguyen and O. Regev. Learning a parallelepiped: cryptanalysis of GGH and NTRU signatures. In S. Vaudenay, editor, *Proc. of Eurocrypt '06*, volume 4004 of LNCS, pages 271–288. Springer, 2006.
15. V. Shoup. Number Theory C++ Library (NTL) version 5.4. Available at `http://www.shoup.net/ntl/`.
16. J. H. Silverman. Invertibility in truncated polynomial rings. Technical report, NTRU Cryptosystems, 2003. Technical reports available at `http://www.ntru.com`.
17. J. H. Silverman and W. Whyte. Technical report n. 18, version 1: Estimating decryption failure probabilities for ntruencrypt. Technical report, NTRU Cryptosystems, 2005.