A Chosen-Ciphertext Attack against NTRU

Éliane Jaulmes[1] and Antoine Joux[2]

[1] SCSSI, 18 rue du Docteur Zamenhof
F-92131 Issy-les-Moulineaux cedex, France
eliane.jaulmes@wanadoo.fr
[2] SCSSI, 18 rue du Docteur Zamenhof
F-92131 Issy-les-Moulineaux cedex, France
Antoine.Joux@ens.fr

Abstract. We present a chosen-ciphertext attack against the public key cryptosystem called NTRU. This cryptosystem is based on polynomial algebra. Its security comes from the interaction of the polynomial mixing system with the independence of reduction modulo two relatively prime integers p and q. In this paper, we examine the effect of feeding special polynomials built from the public key to the decryption algorithm. We are then able to conduct a chosen-ciphertext attack that recovers the secret key from a few ciphertexts/cleartexts pairs with good probability. Finally, we show that the OAEP-like padding proposed for use with NTRU does not protect against this attack.

1 Overview

In [7], Hoffstein, Pipher and Silverman have presented a public key cryptosystem based on polynomial algebra called NTRU. The security of NTRU comes from the interaction of the polynomial mixing system with the independence of reduction modulo p and q. In [7], the authors have studied different possible attacks on their cryptosystem.

First the brute force attack, which can be eased by the meet-in-the-middle principle, may be used against the private key or against a single message. However, for a suitable choice of parameters this attack will not succeed in a reasonable time.

Then there is a multiple transmission attack, which will provide the content of a message that has been transmitted several time. Thus multiple transmissions are not advised. It is also one of the reasons why NTRU recommends a preprocessing scheme.

Finally, several attacks make use of the LLL algorithm of Lenstra-Lenstra-Lovász [10] which produces a reduced basis for a given lattice. They can either recover the secret key from the public key or decipher one given message. However the authors of NTRU claim that the time required is exponential in the degree of the polynomials. For most lattices, it is indeed very difficult to find extremely short vectors. Thus for suitably large degrees, this attack is expected to fail and does fail in practice. Another idea, described by Coppersmith and

Shamir in [3] would be to use LLL to find some short vector in the lattice which could act as a decryption key, but the authors of NTRU claim that experimental evidence suggests that the existence of such spurious keys does not pose a security threat.

However, we show now that it is possible to break the system using a chosen-ciphertext attack. Such attacks have already been used for example in [9] and [5]. They work as follows: The attacker constructs invalid cipher messages. If he can know the plaintexts corresponding to his messages, he can recover some information about the decryption key or even retrieve the private key. In [5], the authors point out that finding the plaintext corresponding to a given ciphertext can reasonably be achieved. This possibility is even increased if decryption is done on a smart card. The standard defense against such attacks is to require redundancy in the message and this is why there exists a padded version of NTRU. The chosen-ciphertext attack we present here has a good probability of recovering the private key from one or two well chosen ciphertexts on the unpadded version of NTRU. It is also able to recover the key on the padded version from a reasonable number of chosen ciphertexts.

This paper is organized as follows: we first recall the main ideas of the cryptosystem without preprocessing, then we present our chosen-ciphertext attack on the unpadded version and give an example of this attack. Finally we study the case where the OAEP-like padding is used and explain how our attack can still recover the private key in this situation.

2 Description of the Cryptosystem

2.1 Notations

The NTRU cryptosystem depends on three integers parameters (N, p, q) and four sets of polynomials of degree $(N-1)$ with integer coefficients, called $\mathcal{L}_f$, $\mathcal{L}_g$, $\mathcal{L}_\phi$, $\mathcal{L}_m$.

The parameters p and q are chosen with $\gcd(p, q) = 1$ and q is much larger than p. All polynomials are in the ring

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

We write $\circledast$ to denote multiplication in R. In the system, some multiplications will be performed modulo q and some modulo p.

The sets $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_\phi$ and $\mathcal{L}_m$ are chosen as follows. The space of messages $\mathcal{L}_m$ consists of all polynomials modulo p. Assuming p is odd, it is most convenient to take

$$\mathcal{L}_m = \left\{ m \in R : \begin{array}{c} m \text{ has coefficients lying between} \\ -\frac{1}{2}(p-1) \text{ and } \frac{1}{2}(p-1) \end{array} \right\}.$$

To describe the other samples spaces, we will use sets of the form

$$\mathcal{L}(d_1, d_2) = \left\{ F \in R : \begin{array}{c} F \text{ has } d_1 \text{ coefficients equal to } 1 \\ d_2 \text{ coefficients equal to } -1, \text{ the rest } 0 \end{array} \right\}.$$

With this notation, we choose three positive integers d_f, d_g, d and set

$$\mathcal{L}_f = \mathcal{L}(d_f, d_f - 1), \quad \mathcal{L}_g = \mathcal{L}(d_g, d_g), \quad \text{and } \mathcal{L}_\phi = \mathcal{L}(d, d).$$

We take $\mathcal{L}_f = \mathcal{L}(d_f, d_f - 1)$ instead of $\mathcal{L}(d_f, d_f)$ because we want f to be invertible and a polynomial satisfying $f(1) = 0$ can never be invertible.

2.2 The Key Generation

To create an NTRU key, one chooses two polynomials $f \in \mathcal{L}_f$ and $g \in \mathcal{L}_g$. The polynomial f must have inverses modulo p and q. We will denote these inverses by F_p and F_q. So we have:

$$F_p \circledast f \equiv 1 \pmod{p} \quad \text{and} \quad F_q \circledast f \equiv 1 \pmod{q}.$$

The public key is then the polynomial:

$$h \equiv F_q \circledast g \pmod{q}.$$

Of course, the parameters N, p, q are public too.
The private key is the polynomial f, together with F_p.

2.3 Encryption and Decryption Procedure

Encryption. The encryption works as follows. First, we select a message m from the set of plaintexts $\mathcal{L}_m$. Next we choose randomly a polynomial $\phi \in \mathcal{L}_\phi$ and use the public key to compute:

$$e \equiv p\phi \circledast h + m \pmod{q}.$$

e is our encrypted message.

Decryption. We have received an encrypted message e and we want to decrypt it using our private key f. To do this, we should have precomputed the polynomial F_p as described in 2.2. In order to decrypt e, we compute :

$$a \equiv f \circledast e \pmod{q},$$

where we choose the coefficients of a in the interval from $-q/2$ to $q/2$. Now, treating a as a polynomial with integer coefficients, we recover the message by computing:

$$F_p \circledast a \pmod{p}.$$

How Decryption Works. The polynomial a verifies

$$\begin{aligned} a \equiv f \circledast e \equiv f \circledast p\phi \circledast h + f \circledast m \pmod{q} \\ = f \circledast p\phi \circledast F_q \circledast g + f \circledast m \pmod{q} \\ = p\phi \circledast g + f \circledast m \pmod{q}. \end{aligned}$$

For appropriate parameter choices, we can ensure that all coefficients of the polynomial $p\phi \circledast g + f \circledast m$ lie between $-q/2$ and $q/2$. So the intermediate value $p\phi \circledast g + f \circledast m \bmod q$ is in fact the true (non modular) value of this polynomial. This means that when we compute a and reduce its coefficients into this interval, we recover exactly the polynomial $p\phi \circledast g + f \circledast m$. Hence its reduction modulo p give us $f \circledast m \bmod p$ and the multiplication by F_p retrieves the message m.

The basic idea for the attack presented here will be to construct intermediate polynomials such that the modular values differ from the true values.

2.4 Sets of Parameters for NTRU

The authors of NTRU have defined different sets of parameters for NTRU providing various security levels. Theses parameters are given in [12].

Name	N	p	q	$\mathcal{L}_f$	$\mathcal{L}_g$	$\mathcal{L}_\phi$
Case A	107	3	64	$\mathcal{L}(15,14)$	$\mathcal{L}(12,12)$	$\mathcal{L}(5,5)$
Case B	167	3	128	$\mathcal{L}(61,60)$	$\mathcal{L}(20,20)$	$\mathcal{L}(18,18)$
Case C	263	3	128	$\mathcal{L}(50,49)$	$\mathcal{L}(24,24)$	$\mathcal{L}(16,16)$
Case D	503	3	256	$\mathcal{L}(216,215)$	$\mathcal{L}(72,72)$	$\mathcal{L}(55,55)$

In the original formulation of the NTRU public key cryptosystem [7], it was suggested that one could use $N = 107$ to create a cryptosystem with moderate security. Such a system can be broken by lattice attacks in a few hours. Thus the use of case A is not recommended anymore but we will still use it to describe our attack in its simple version.

3 The Chosen-Ciphertext Attack

3.1 Principle

As stated in 2.3, we want to build cipher texts such that the intermediate values in the deciphering process will differ from the true values. We first consider the effect of deciphering a cipher text of the form $ch + c$, where c is an integer and h is the public key. The decryption algorithm first multiplies by f modulo q:

$$\begin{aligned} a \equiv f \circledast ch + cf \pmod{q} \\ \equiv cg + cf \pmod{q}, \end{aligned}$$

where g and f both have coefficients equal to 0, 1 or -1. Hence the polynomial $cf + cg$ have coefficients equal to 0, c, $-c$, $2c$ or $-2c$. We then need to reduce the coefficients of a between $-q/2$ and $q/2$. If c has been chosen such that $c < q/2$ and $2c > q/2$, we will have to reduce only the coefficients equal to $2c$ or $-2c$.

If we now suppose that a single coefficient in a is $\pm 2c$, say $a_i = +2c$, then the value of a mod q is $cg + cf - qx^i$. The deciphering process outputs

$$cg \circledast F_p + c - qx^i \circledast F_p \pmod p$$

If c has been chosen as a multiple of p, then the output is

$$-qx^i \circledast F_p \pmod p.$$

Since $\gcd(p,q) = 1$, we can recover $x^i \circledast F_p \equiv x^i/f \bmod p$ and compute its inverse $f/x^i \bmod p$. Since all the coefficients of f are 1 or -1, it is the true value of the polynomial. We can then compute

$$g/x^i = h \circledast f/x^i \pmod q,$$

which is also the true value of g/x^i. Going back to the key process described in section 2.2, we can see that (f,g) and $(f/x^i, g/x^i)$ are equivalent keys.

Of course, in general, the polynomial $cf + cg$ may have none or several coefficients equal to $\pm 2c$, and then the above attack does not work anymore. In the next section, we will analyze the attack and generalize it to make it work for all the security parameters proposed for NTRU in [7].

3.2 Analysis of the Attack

We say that two polynomials P_1 and P_2 have a *collision* when they have the same non zero coefficient at the same degree.

We now define the *intersection* polynomial k of (P_1, P_2) by:

$$k = \sum k_i x^i,$$

where

$$k_i = \begin{cases} 1 \text{ if } P_1 \text{ and } P_2 \text{ both have their } i^{\text{th}} \text{ coefficient equal to } 1 \\ -1 \text{ if } P_1 \text{ and } P_2 \text{ both have their } i^{\text{th}} \text{ coefficient equal to -1} \\ 0 \text{ otherwise} \end{cases}$$

Using this notation, we write again the result of the first decryption step of $c + ch$, as seen in section 3.1. $a \equiv cg + cf \bmod q = c + ch - qk$

The decrypted message obtained is then

$$\begin{aligned} m &\equiv cF_p \circledast f + cF_p \circledast g - qF_p \circledast k \pmod p \\ &\equiv c + ch - qF_p \circledast k \pmod p \end{aligned}$$

Since c has been chosen such that $c \equiv 0 \bmod p$,

$$m = -qF_p \circledast k \pmod p.$$

The private key f can then be obtained from $f \equiv -qk \circledast m^{-1} \bmod p$

When f and g have few common coefficients, the polynomial k has only a few non zero coefficients. By testing different values for k, we can compute possible polynomials f. The private key is likely the one that satisfies the condition $f \in \mathcal{L}_f$. It is then a simple matter to verify our guess by trying to decrypt a message with f or by computing $h \circledast f \bmod q = g'$. Then if $g' = \pm x^i \circledast g$, we know we have a correct key.

Let us study the probability of success of our attack over the sets of parameters given in section 2.4.

The probability of f and g having one and only one collision is the following:

$$p = p_1 + p_{-1},$$

where p_1, the probability of collision of two 1, is:

$$\sum_{k=0}^{min(d_f-1,d_g)} \frac{\binom{d_g}{1}\binom{d_g}{k}\binom{N-2d_g}{d_f-1-k}\binom{N-d_f-d_g+k}{d_f-1}}{\binom{N}{d_f}\binom{N-d_f+1}{d_f-1}}$$

and p_{-1}, the probability of collision of two -1, is:

$$\sum_{k=0}^{min(d_f-2,d_g)} \frac{\binom{d_g}{1}\binom{d_g}{k}\binom{N-2d_g}{d_f-2-k}\binom{N-d_f+1-d_g+k}{d_f}}{\binom{N}{d_f}\binom{N-d_f+1}{d_f-1}}$$

There are similar formulas for more collisions. However, they are quickly cumbersome to compute.

Another approach is to evaluate the expected number of collisions between f and g. An heuristic approximation of this number is

$$\frac{(2d_f-1)d_g}{N}.$$

In case A, we find an average number of collisions of 3.25. We can thus expect k to have around three non zero coefficients.

The table below shows the different probabilities of collisions in the different proposed cases. It also gives the average expected number of collisions.

	Case A	Case B	Case C	Case D
Average number of collisions	3.25	14.5	9.03	61.7
Probability of 0 collision	0.026	$9.3 * 10^{-9}$	$3.1 * 10^{-5}$	$2 * 10^{-36}$
Probability of 1 collision	0.13	$5.8 * 10^{-7}$	$5 * 10^{-4}$	$1.1 * 10^{-33}$
Probability of 2 collisions	0.25	$9.5 * 10^{-6}$	$3 * 10^{-3}$	$1.5 * 10^{-31}$
Probability of 3 collisions	0.28	$8.6 * 10^{-5}$	0.011	$1.2 * 10^{-29}$
Probability of 4 collisions	0.22	$5.1 * 10^{-4}$	0.028	$7.3 * 10^{-28}$

For example, with the parameters of NTRU 107, which has a key security of 2^{50} against a meet-in-the-middle attack, we have a one-collision probability of $p = 0.13$. It means one over ten cipher messages will produce a polynomial k with a single non zero coefficient and the simple case described in section 3.1 will apply. We can see that the attack, as it has currently been described, will fail in cases B, C and D. In section 3.3, we generalize our idea to make it work in those cases.

In general, k may have more than one coefficient, and we need to enumerate the possible k and compute $f' = k/m \bmod p$, where m is our decrypted message. When $f' \in \mathcal{L}_f$, we have found a likely polynomial. We just need to verify that f' is able to decrypt messages. If we now analyze the number of possible polynomials k we need to test in order to recover the private key, we can first note that the polynomials of the form $x^i f \bmod x^N - 1$ have as many coefficients equal to 1 and -1 as f. As the multiplication by x^i will not change the value of the coefficients of a and as the decryption proceeding consists in multiplying and dividing by f, the rotated key $f' = x^i f \bmod x^N - 1$ can be used to decrypt any message encrypted with f. Hence we can assume $k(0) \neq 0$.

So if we assume that k has n non zero coefficients, we will have to try $2^n \binom{N-1}{n-1}$ different values for k.

We can see in the table below the approximate number of polynomials we need to test function of the expected number of collisions.

Expected no of collisions	Case A	Case B	Case C	Case D
1 collision	2	2	2	2
2 collisions	2^9	2^{10}	2^{10}	2^{11}
3 collisions	2^{16}	2^{17}	2^{18}	2^{20}
4 collisions	2^{22}	2^{24}	2^{26}	2^{29}

The message $c + ch$ can fail to produce the private key, if f and g have too many collisions. We can then try again with $cx + ch$ and more generally with polynomials of the form $cx^i + ch$. This means considering collisions between g and $x^i f$ mod $x^N - 1$. So there is a compromise between the number of possible collisions we will test and the number of cipher texts we will need. Many ciphertexts are likely to produce at least a polynomial whose number of non zero coefficient is below the average value. If we have only one ciphertext, it may take more time to test possible polynomials before finding the key.

3.3 Extending to Higher Security Parameters

As seen in section 3.2, the parameters proposed in [7] for higher security give us a very high number of collisions. This means that there will be an extremely low probability of having only a few collisions. Therefore, we can no longer use messages of the form cx^i+ch. Instead, we reduce the average number of collisions by testing messages of the form

$$chx^{i_1} + \cdots + chx^{i_n} + cx^{j_1} + cx^{j_2} + \cdots + cx^{j_m},$$

where c is a multiple of p that verifies

$$(n+m-1)c < q/2 \quad \text{and} \quad (n+m)c > q/2.$$

We choose the numbers n and m in order to get a good probability of having only one or two collisions. As before, we do not explicitely compute these probabilities, but we estimate the average number of collisions. When this number is near 1, it means that the n and m are correctly chosen. An heuristic approximation of the number of collisions is given by:

$$\frac{2d_f^m d_g^n}{N^{n+m-1}}$$

4 Example

4.1 Detailed Example of Case D

In [7], it is claimed that the highest security level will be obtained with the set of parameters D.

We now give an example that shows, with this set of parameters, that our attack can recover the secret key.

Here is the private key (f, g) we have used:

$$\begin{aligned} f = & -x^{502} + x^{501} + x^{500} - x^{499} - x^{498} + x^{497} - x^{496} - x^{495} - x^{494} - x^{493} - x^{492} \\ & -x^{491} + x^{490} - x^{488} + x^{487} - x^{486} - x^{485} - x^{482} + x^{481} - x^{480} - x^{479} + x^{477} \\ & +x^{475} + x^{474} + x^{472} - x^{471} + x^{470} + x^{468} - x^{467} + x^{466} + x^{464} - x^{463} + x^{462} \\ & +x^{461} - x^{460} - x^{459} + x^{458} + x^{457} - x^{455} + x^{454} - x^{453} - x^{451} - x^{450} + x^{449} \\ & +x^{448} + x^{447} + x^{446} + x^{445} - x^{444} - x^{443} + x^{442} + x^{441} + x^{440} - x^{439} + x^{438} \\ & -x^{437} - x^{436} - x^{435} + x^{434} + x^{433} - x^{430} - x^{429} + x^{428} - x^{425} + x^{424} + x^{423} \\ & -x^{422} - x^{421} - x^{420} - x^{418} - x^{417} - x^{416} + x^{415} - x^{414} + x^{412} - x^{411} - x^{409} \\ & -x^{408} + x^{407} + x^{406} - x^{405} + x^{404} - x^{402} - x^{401} - x^{400} + x^{399} - x^{398} + x^{397} \end{aligned}$$

$$
\begin{aligned}
&+x^{396} - x^{394} + x^{393} - x^{391} + x^{390} + x^{389} + x^{388} - x^{387} - x^{386} + x^{385} + x^{384} \\
&+x^{383} + x^{381} - x^{380} - x^{379} + x^{378} - x^{377} + x^{376} + x^{374} - x^{373} + x^{372} + x^{371} \\
&+x^{370} - x^{369} + x^{368} - x^{367} + x^{366} - x^{365} + x^{364} - x^{363} - x^{362} + x^{361} - x^{360} \\
&+x^{359} - x^{358} - x^{357} + x^{356} + x^{355} - x^{354} + x^{353} - x^{352} + x^{350} - x^{349} - x^{348} \\
&-x^{346} - x^{345} + x^{344} - x^{343} - x^{342} + x^{341} - x^{340} - x^{339} - x^{338} + x^{337} + x^{336} \\
&+x^{334} + x^{333} - x^{332} - x^{331} - x^{330} + x^{329} - x^{328} + x^{327} + x^{326} + x^{325} - x^{324} \\
&+x^{323} - x^{322} + x^{321} - x^{320} - x^{319} + x^{318} + x^{317} + x^{316} - x^{315} - x^{313} - x^{311} \\
&-x^{310} - x^{309} + x^{308} - x^{306} - x^{305} + x^{304} - x^{303} + x^{302} - x^{301} + x^{300} - x^{299} \\
&-x^{298} - x^{297} + x^{294} - x^{293} - x^{292} - x^{291} - x^{290} - x^{288} + x^{287} - x^{286} - x^{285} \\
&+x^{284} - x^{283} + x^{282} - x^{280} - x^{279} + x^{277} - x^{276} + x^{275} + x^{274} + x^{273} + x^{272} \\
&-x^{271} + x^{270} - x^{269} + x^{268} - x^{267} - x^{266} + x^{264} + x^{263} - x^{262} + x^{261} - x^{260} \\
&+x^{259} - x^{257} + x^{256} - x^{255} - x^{254} + x^{253} + x^{252} + x^{251} + x^{249} + x^{248} - x^{247} \\
&+x^{246} - x^{245} + x^{243} - x^{242} + x^{240} + x^{238} - x^{237} - x^{236} + x^{234} - x^{233} - x^{232} \\
&+x^{231} - x^{230} + x^{229} - x^{228} - x^{227} + x^{226} - x^{225} + x^{223} + x^{222} - x^{221} + x^{220} \\
&+x^{219} + x^{218} - x^{217} - x^{215} - x^{214} + x^{213} - x^{212} + x^{210} - x^{209} + x^{208} + x^{207} \\
&-x^{206} - x^{205} + x^{203} + x^{202} - x^{201} - x^{200} + x^{199} + x^{198} - x^{197} + x^{196} + x^{195} \\
&-x^{194} + x^{193} + x^{192} + x^{191} + x^{190} + x^{188} + x^{187} - x^{186} + x^{185} - x^{184} + x^{183} \\
&+x^{182} + x^{181} + x^{180} - x^{179} - x^{178} + x^{177} - x^{176} + x^{175} + x^{174} - x^{173} + x^{172} \\
&-x^{170} + x^{169} + x^{168} + x^{167} + x^{166} - x^{165} - x^{164} + x^{161} + x^{160} - x^{159} + x^{158} \\
&-x^{155} + x^{154} + x^{152} + x^{151} - x^{150} + x^{149} + x^{148} + x^{147} - x^{145} - x^{142} + x^{141} \\
&-x^{140} - x^{139} + x^{138} + x^{137} - x^{136} - x^{135} + x^{133} - x^{132} + x^{131} + x^{130} + x^{128} \\
&+x^{127} - x^{126} + x^{125} + x^{124} + x^{123} - x^{121} + x^{120} + x^{118} - x^{116} + x^{115} - x^{114} \\
&-x^{113} - x^{112} + x^{110} + x^{109} + x^{108} + x^{107} - x^{106} - x^{105} - x^{103} + x^{102} + x^{100} \\
&+x^{99} + x^{98} + x^{96} + x^{95} - x^{94} - x^{93} - x^{92} + x^{91} - x^{90} - x^{89} - x^{88} - x^{87} \\
&+x^{86} - x^{85} + x^{84} + x^{83} - x^{82} + x^{81} - x^{80} + x^{79} + x^{78} + x^{77} + x^{75} + x^{74} \\
&-x^{73} - x^{72} - x^{71} - x^{69} - x^{68} - x^{67} + x^{66} + x^{65} + x^{64} + x^{63} + x^{62} - x^{60} \\
&-x^{59} + x^{58} - x^{57} + x^{56} + x^{55} + x^{54} + x^{53} - x^{51} - x^{50} + x^{49} + x^{48} - x^{47} \\
&+x^{46} + x^{45} + x^{44} - x^{43} - x^{42} + x^{41} + x^{40} - x^{39} - x^{38} + x^{37} - x^{36} + x^{35} \\
&-x^{34} - x^{32} - x^{31} + x^{30} - x^{29} - x^{28} + x^{27} - x^{25} - x^{24} - x^{23} - x^{21} + x^{20} \\
&-x^{19} + x^{18} - x^{17} - x^{16} - x^{15} - x^{14} + x^{13} + x^{12} - x^{11} - x^{10} + x^{9} - x^{8} \\
&-x^{7} - x^{6} - x^{5} - x^{3} + x^{2} - 1
\end{aligned}
$$

$$
\begin{aligned}
g = &-x^{499} + x^{496} + x^{495} - x^{487} + x^{486} + x^{484} - x^{480} + x^{478} + x^{470} - x^{466} + x^{465} \\
&-x^{462} + x^{461} + x^{460} + x^{451} - x^{446} - x^{431} - x^{428} + x^{421} + x^{415} + x^{412} - x^{411} \\
&-x^{406} - x^{403} - x^{402} - x^{398} - x^{397} - x^{395} + x^{392} + x^{373} - x^{371} - x^{370} + x^{367} \\
&+x^{366} - x^{364} - x^{359} - x^{355} + x^{352} + x^{351} + x^{349} + x^{347} + x^{340} + x^{339} + x^{338} \\
&+x^{335} + x^{328} - x^{326} + x^{323} + x^{317} - x^{314} - x^{309} - x^{308} + x^{307} + x^{306} + x^{304} \\
&-x^{303} - x^{302} - x^{299} - x^{295} - x^{292} + x^{291} + x^{289} + x^{288} + x^{283} + x^{281} + x^{280} \\
&-x^{277} + x^{266} + x^{264} - x^{262} - x^{260} - x^{257} + x^{256} - x^{255} - x^{251} - x^{250} - x^{249} \\
&-x^{236} - x^{235} + x^{233} - x^{232} + x^{230} + x^{227} + x^{226} - x^{224} + x^{217} + x^{216} - x^{215} \\
&-x^{212} + x^{206} - x^{205} + x^{203} + x^{196} - x^{194} + x^{193} + x^{190} + x^{185} - x^{183} - x^{177} \\
&-x^{172} - x^{169} - x^{168} + x^{165} - x^{163} - x^{157} + x^{156} + x^{155} - x^{138} + x^{136} - x^{135} \\
&+x^{134} + x^{132} - x^{131} - x^{123} + x^{119} - x^{117} - x^{111} - x^{102} - x^{99} + x^{97} - x^{95} \\
&-x^{94} + x^{92} + x^{91} - x^{89} - x^{88} - x^{86} + x^{84} + x^{83} - x^{78} + x^{76} - x^{66} + x^{60} \\
&-x^{52} + x^{51} - x^{47} + x^{46} - x^{36} - x^{35} - x^{34} + x^{30} + x^{28} + x^{16} + 1
\end{aligned}
$$

We do not give here values of F_p, F_q or of the public key h since they are big and they can easily be computed from f and g.

If we use messages of the form $c + chx^{i_1} + chx^{i_2} + chx^{i_3}$, our heuristic estimates the average number of collisions by 1.26.

We want c to verify $c \bmod p = 0$, $3c < q/2$ and $4c > q/2$. We chose $c = 33$, which satisfies this conditions.

We use the chosen ciphertext $e = 33h + 33 + 33hx + 33hx^4$.

Let m be the decoded message. We find then that

$$(1 + x^{67})/m \pmod p$$

is a possible value f'.

That gives us the following value for f'

$$\begin{aligned}
f' = {} & x^{501} - x^{500} - x^{499} + x^{498} - x^{497} + x^{496} - x^{495} - x^{494} + x^{493} - x^{492} + x^{490} \\
& + x^{489} - x^{488} + x^{487} + x^{486} + x^{485} - x^{484} - x^{482} - x^{481} + x^{480} - x^{479} + x^{477} \\
& - x^{476} + x^{475} + x^{474} - x^{473} - x^{472} + x^{470} + x^{469} - x^{468} - x^{467} + x^{466} + x^{465} \\
& - x^{464} + x^{463} + x^{462} - x^{461} + x^{460} + x^{459} + x^{458} + x^{457} + x^{455} + x^{454} - x^{453} \\
& + x^{452} - x^{451} + x^{450} + x^{449} + x^{448} + x^{447} - x^{446} - x^{445} + x^{444} - x^{443} + x^{442} \\
& + x^{441} - x^{440} + x^{439} - x^{437} + x^{436} + x^{435} + x^{434} + x^{433} - x^{432} - x^{431} + x^{428} \\
& + x^{427} - x^{426} + x^{425} - x^{422} + x^{421} + x^{419} + x^{418} - x^{417} + x^{416} + x^{415} + x^{414} \\
& - x^{412} - x^{409} + x^{408} - x^{407} - x^{406} + x^{405} + x^{404} - x^{403} - x^{402} + x^{400} - x^{399} \\
& + x^{398} + x^{397} + x^{395} + x^{394} - x^{393} + x^{392} + x^{391} + x^{390} - x^{388} + x^{387} + x^{385} \\
& - x^{383} + x^{382} - x^{381} - x^{380} - x^{379} + x^{377} + x^{376} + x^{375} + x^{374} - x^{373} - x^{372} \\
& - x^{370} + x^{369} + x^{367} + x^{366} + x^{365} + x^{363} + x^{362} - x^{361} - x^{360} - x^{359} + x^{358} \\
& - x^{357} - x^{356} - x^{355} - x^{354} + x^{353} - x^{352} + x^{351} + x^{350} - x^{349} + x^{348} - x^{347} \\
& + x^{346} + x^{345} + x^{344} + x^{342} + x^{341} - x^{340} - x^{339} - x^{338} - x^{336} - x^{335} - x^{334} \\
& + x^{333} + x^{332} + x^{331} + x^{330} + x^{329} - x^{327} - x^{326} + x^{325} - x^{324} + x^{323} + x^{322} \\
& + x^{321} + x^{320} - x^{318} - x^{317} + x^{316} + x^{315} - x^{314} + x^{313} + x^{312} + x^{311} - x^{310} \\
& - x^{309} + x^{308} + x^{307} - x^{306} - x^{305} + x^{304} - x^{303} + x^{302} - x^{301} - x^{299} - x^{298} \\
& + x^{297} - x^{296} - x^{295} + x^{294} - x^{292} - x^{291} - x^{290} - x^{288} + x^{287} - x^{286} + x^{285} \\
& - x^{284} - x^{283} - x^{282} - x^{281} + x^{280} + x^{279} - x^{278} - x^{277} + x^{276} - x^{275} - x^{274} \\
& - x^{273} - x^{272} - x^{270} + x^{269} - x^{267} - x^{266} + x^{265} + x^{264} - x^{263} - x^{262} + x^{261} \\
& - x^{260} - x^{259} - x^{258} - x^{257} - x^{256} - x^{255} + x^{254} - x^{252} + x^{251} - x^{250} - x^{249} \\
& - x^{246} + x^{245} - x^{244} - x^{243} + x^{241} + x^{239} + x^{238} + x^{236} - x^{235} + x^{234} + x^{232} \\
& - x^{231} + x^{230} + x^{228} - x^{227} + x^{226} + x^{225} - x^{224} - x^{223} + x^{222} + x^{221} - x^{219} \\
& + x^{218} - x^{217} - x^{215} - x^{214} + x^{213} + x^{212} + x^{211} + x^{210} + x^{209} - x^{208} - x^{207} \\
& + x^{206} + x^{205} + x^{204} - x^{203} + x^{202} - x^{201} - x^{200} - x^{199} + x^{198} + x^{197} - x^{194} \\
& - x^{193} + x^{192} - x^{189} + x^{188} + x^{187} - x^{186} - x^{185} - x^{184} - x^{182} - x^{181} - x^{180} \\
& + x^{179} - x^{178} + x^{176} - x^{175} - x^{173} - x^{172} + x^{171} + x^{170} - x^{169} + x^{168} - x^{166} \\
& - x^{165} - x^{164} + x^{163} - x^{162} + x^{161} + x^{160} - x^{158} + x^{157} - x^{155} + x^{154} + x^{153} \\
& + x^{152} - x^{151} - x^{150} + x^{149} + x^{148} + x^{147} + x^{145} - x^{144} - x^{143} + x^{142} - x^{141} \\
& + x^{140} + x^{138} - x^{137} + x^{136} + x^{135} + x^{134} - x^{133} + x^{132} - x^{131} + x^{130} - x^{129} \\
& + x^{128} - x^{127} - x^{126} + x^{125} - x^{124} + x^{123} - x^{122} - x^{121} + x^{120} + x^{119} - x^{118} \\
& + x^{117} - x^{116} + x^{114} - x^{113} - x^{112} - x^{110} - x^{109} + x^{108} - x^{107} - x^{106} + x^{105} \\
& - x^{104} - x^{103} - x^{102} + x^{101} + x^{100} + x^{98} + x^{97} - x^{96} - x^{95} - x^{94} + x^{93} \\
& - x^{92} + x^{91} + x^{90} + x^{89} - x^{88} + x^{87} - x^{86} + x^{85} - x^{84} - x^{83} + x^{82} + x^{81} \\
& + x^{80} - x^{79} - x^{77} - x^{75} - x^{74} - x^{73} + x^{72} - x^{70} - x^{69} + x^{68} - x^{67} + x^{66} \\
& - x^{65} + x^{64} - x^{63} - x^{62} - x^{61} + x^{58} - x^{57} - x^{56} - x^{55} - x^{54} - x^{52} + x^{51} \\
& - x^{50} - x^{49} + x^{48} - x^{47} + x^{46} - x^{44} - x^{43} + x^{41} - x^{40} + x^{39} + x^{38} + x^{37} \\
& + x^{36} - x^{35} + x^{34} - x^{33} + x^{32} - x^{31} - x^{30} + x^{28} + x^{27} - x^{26} + x^{25} - x^{24} \\
& + x^{23} - x^{21} + x^{20} - x^{19} - x^{18} + x^{17} + x^{16} + x^{15} + x^{13} + x^{12} - x^{11} + x^{10} \\
& - x^{9} + x^{7} - x^{6} + x^{4} + x^{2} - x - 1
\end{aligned}$$

This value is different from the original one (we have $f = x^{236} \circledast f'$), but it can be used to decrypt messages nonetheless.

4.2 Choice of Parameters and Running Times Table

Here we give estimation of the running times for the different sets of parameters and the values chosen for m, n and c.

Case	A	B		C	D	
m	1	4	1	1	4	1
n	1	1	2	2	2	3
c	18	15	24	24	24	33
Avg no of collisions	3.36	0.712	1.75	0.832	0.7	1.27
No of ciphertexts (testing 1 collision)	–	4.5	–	2.2	2.25	–
No of ciphertexts (testing 2 collisions)	7	–	2	2	–	2
Time to test for 1 collision	–	1s	–	6s	85s	–
Time to test for 2 collisions	25s	–	135s	4mn	–	1h

These running times have been obtain on a single PC, using `GP/PARI CALCULATOR Version 2.0.14`.

5 Plaintext Awareness with Our Chosen-Ciphertext Attack

The attack described in the previous sections uses the fact that one can build a ciphertext without knowing the corresponding plaintext. A cryptosystem is said to be plaintext aware if it is infeasible for an attacker to construct a valid ciphertext without knowing the corresponding plaintext (see [2] which first introduced this notion and [1] which had a corrected definition). So in [11] Silverman proposed to use a system similar to OAEP to make NTRU plaintext aware. OAEP stands for Optimal Asymmetric Encryption Padding. It has been proposed by Mihir Bellare and Phillip Rogaway in [2] and describes an embedding scheme using an hash and a generating function that achieves plaintext-aware encryption. However, since OAEP applies only to a one-way trapdoor function, it had to be adapted to work for NTRU.

5.1 A Description of the Embedding Scheme Proposed for NTRU

We let

$$\mathcal{P}_p(N) = \{\text{polynomials of degree at most } N-1 \text{ with mod } p \text{ coefficients}\},$$

and we write

$$[g]_p = \begin{cases} g \text{ with its coefficients reduced} \\ \text{modulo } p \text{ into the range }]-p/2, p/2]. \end{cases}$$

We need a generating function and a hash function

$$G : \mathcal{P}_p(N) \to \mathcal{P}_p(N) \text{ and } H : \mathcal{P}_p(N) \times \mathcal{P}_p(N) \to \mathcal{P}_p(K).$$

To encrypt a message, one chooses a plaintext m from the set of plaintexts $\mathcal{P}_p(N-K)$ and a polynomial $\phi \in \mathcal{L}_\phi$. One computes

$$e \equiv p\phi \circledast h + [m + H(m, [p\phi \circledast h]_p)X^{N-K} + G([p\phi \circledast h]_p)]_p \pmod{q}. \qquad (1)$$

To decrypt the message, the receiver uses his private key f and the standard NTRU decryption method to recover a polynomial

$$n = [F_p \circledast [f \circledast e]_q]_p \in \mathcal{P}_p(N).$$

Next he computes

$$b \equiv e - n \pmod{p} \qquad \text{and} \qquad c \equiv n - G(b) \pmod{p}.$$

and he writes c in the form

$$c = c' + c'' X^{N-K} \text{ with } \deg(c') < N-K \text{ and } \deg(c'') < K.$$

Finally, he compares the quantities

$$c'' \text{ and } H(c', b).$$

If they are the same, he accepts c' as a valid decryption. Otherwise he rejects the message as invalid.

An attacker who does not know the underlying plaintext of a cipher message will have a probability of p^{-K} of producing a valid ciphertext.

We are now going to show how our attack is modified with this encapsulation.

5.2 Adaptation of Our Attack

Principle. With this embedding, an attacker can detect when a message is valid or invalid. Our goal is to produce special messages that may be either valid or invalid and learn information from their acceptance or rejection.

As in the unpadded version, this is achieved by replacing $p\phi \circledast h$ by a well chosen polynomial. We add to this polynomial the correct encapsulation of a message m, so that the ciphertext will be accepted when there is no collision in the polynomial and rejected otherwise.

The principle of our attack is close to what Hall, Goldberg and Schneier call a *reaction attack* in [6]. It is a chosen-ciphertext attack but does not require that the attacker sees the decrypted plaintext. He only needs to know whether the ciphertext was correctly decrypted or rejected for errors.

Such attacks have been studied on NTRU by Hoffstein and Silverman in [8] but they applied on the unpadded version of the cryptosystem.

Choice of a Polynomial P. Let

$$P \equiv x^{i_1} + \cdots + x^{i_n} + h \circledast (x^{j_1} + \cdots x^{j_m}) \pmod{q}, \qquad i_k, j_l \in \mathbb{N}.$$

and choose n and m such that the average number of collisions, as defined in section 3.3, in P is near 1, and preferably a little smaller, so that we can expect P to have no more than one collision. If there is no collision, there will be no decryption failure, and we will know we need to change P. We will have to try different P, till we found a suitable one.

Now, since multiplying by $\pm x^i$ does not change the propriety of f and h to act as private and public key, we can assume the collision happens at degree 0 and is a collision of 1. This will simplify the presentation of the attack.

Information Obtained from Decryption Failure. Now if we can ask the decryption of messages of the form $cx^i + cP$, for i ranging from 0 to $N-1$, with c such that $c \equiv 0 \bmod P$, $(n+m)c < q/2$ and $(n+m+1)c > q/2$, we can discover all coefficients equal to 1 in f. Indeed let us assume that we send a message of the above form and that we expect the decrypted message to be 0. If the answer of the decryption is not 0, then the decryption process will send an error since we cannot know the plaintext.

Now, as we have seen in section 3.1, decryption will be different from 0 if and if only there is collision between the $(N-i)$th coefficient of f and the unique collision in P. So if decryption is 0, the $(N-i)$th coefficient in f will be a 0 or a -1 and if decryption is different than 0, that is if we have a decryption error, we know that the $(N-i)$th coefficient of f is a 1. Similarly, with messages $cx^i - cP$, a decryption error indicates that the $(N-i)$th coefficient of f is a -1. By testing those $2N$ messages, we can reconstruct a key f' equivalent to f.

Influence of the Encapsulation. But, as stated above, we now have to add some valid encapsulated message to our test cipher $cx^i \pm cP$ (otherwise all our test messages will be rejected and we will not learn anything), so we do not send $cx^i \pm cP$, but $cx^i \pm cP + m'$. The message m' can be chosen as the correct encapsulation of any message m, where $p\phi \circledast h$ has been replaced by $cx^i \pm cP$ in the formula (1).

After multiplication by f, we obtain $cx^i \circledast f \pm cP \circledast f + m' \circledast f$. The coefficients of $m' \circledast f$ may be of size $q/4$ and thus can produce a wrong decryption where we should have had a good one according only to $cx^i \circledast f \pm cP \circledast f$. It is not possible to get rid of the influence of $m' \circledast f$, but we can reduce it. It is indeed possible to take for m the value $-G([cx^i \pm cP]_p) \bmod p$ truncated to degree $N-K$, so that $m' = [m + H(m, [cx^i \pm cP]_p)X^{N-k} + G([cx^i \pm cP]_p)]_p$ will have all its coefficients of degree less than $N-K$ equal to zero. m' has now only approximately $2K/3$ non zero coefficients, and $m' \circledast f$ will have coefficients whose absolute value may be less than $\min((5c - q/2), (q/2 - 4c))$. Then hopefully $cP + m'$ will have the same property than cP, that is produce a wrong message when added to cx^i if and if only the $(N-i)$th coefficient of f is 1. Note that if $cP+m'$ verify this, we can proceed exactly as described above to recover the private key. The problem is that m' should be recalculated each time, for each value of $cx^i \pm cP$. But, since m' comes from $[[cP]_q]_p$, let us see what happens when we add cx^i to cP: in the majority of cases, the addition of cx^i to $[cP]_q$ will not induce a new reduction modulo q so that $[[cP]_q + cx^i]_p = [[cP]_q]_p$ (recall that $c \equiv 0 \bmod p$), and m' will stay the same. For such i, we can use the system described above to determine the corresponding coefficients of f. For the other coefficients, we cannot be really sure of the coefficients we obtain, even if there is a good probability for them to be right. It is then possible to use either LLL algorithm to find the missing coefficients or choose another value for P and repeat the process.

5.3 Example

Algorithm. We give first a brief description of the resulting algorithm to attack NTRU.

1. Choose appropriate values for m and n such that the heuristic number of collisions $\frac{2d_f^n d_g^m}{N^{n+m-1}}$ will be near 1.
2. Select a suitable c with $c \equiv 0 \bmod p$, $(m+n)c < q/2$ and $(m+n+1)c > q/2$.

3. Select a value of a polynomial P.

$$P = x^{i_1} + \cdots + x^{i_n} + h \circledast (x^{j_1} + \cdots + x^{j_m}) \pmod q$$

4. Produce m' corresponding to cP: $m' = [m + H(m, [cP]_p)X^{N-k} + G([cP]_p)]_p$ with $m = [-G([cP]_p)]_p \bmod X^{N-K}$.
5. Ask the decryption of $cP + m'$. The answer should be m. If not, go back to 3.
6. For all i such that $[[cx^i + cP]_q]_p = [[cP]_q]_p$, ask decryption of $cx^i + cP + m'$. If the answer is a decryption error, the (N-i)th coefficient of f' is a 1, else we know it is *not* a 1. For all other i, the (N-i)th coefficient of f' may be a 1.
7. At the same time, for all i such that $[[-cx^i + cP]_q]_p = [[cP]_q]_p$, ask decryption of $-cx^i + cP + m'$. If the answer is a decryption error, the (N-i)th coefficient of f' is a -1, else we know it is *not* a -1. Note that if we had $[[cx^i + cP]_q]_p \neq [[cP]_q]_p$, then $[[-cx^i + cP]_q]_p = [[cP]_q]_p$. So a coefficient can not both possibly be a 1 and -1.
8. Note also that if $cx^i + cP + m'$ gave a decryption error, then $-cx^i + cP + m'$ should not. If this is the case, we know that m' introduced decryption errors and we go back to step 3.
9. If after a few messages there is still no decryption failure, there is no collision in P. Go back to step 3.
10. Count the minimal and maximal number of 1 and -1 in f'. If this number is not consistent with the value of d_f, go back to step 3.
11. Merge with preceeding informations obtained on f'. Eventually repeat with another P (step 3).

Application. Here is an example of the attack with the following set of parameters:

- $(N, p, q) = (503, 3, 256)$
- $n_f = 216$
- $n_g = 72$
- $K = 107$

Those are the parameters proposed in [11] to offer the highest security.
For $n = 1$ and $m = 3$, we find an average number of collisions equal to 1.267.
We want $c \equiv 0 \bmod 3$, $4c < 128$, $5c > 128$. We choose $c = 27$.
We tested the following polynomials P:

- $P = 1 + h \circledast (x + x^2 + x^3)$
- $P = 1 + h \circledast (x + x^2 + x^4)$
- $P = 1 + h \circledast (x + x^2 + x^5)$
- $P = 1 + h \circledast (x + x^2 + x^6)$
- $P = 1 + h \circledast (x + x^2 + x^7)$

The good ones where:

- $P = 1 + h \circledast (x + x^2 + x^4)$
- $P = 1 + h \circledast (x + x^2 + x^7)$

The other ones failed at step 8 or 9.

After merging the informations gained from these two polynomials, we had only 15 possible keys left. It is then easy to find the good one by trying to decipher a ciphertext or by testing whether $h \circledast f' \equiv \pm x^i \circledast g \bmod q$ for some i.

We were able to recover the private key with less than $5N$ calls to the decryption oracle.

We give now a few statistics of our algorithm with the different sets of parameters.

Case	A	B	B	C	D
Value for K	17	49	49	65	107
Avg no of ciphertexts	230	310	620	950	2100
Avg running time	11s	17mn	2mn	6mn	36mn

Remark: even for the highest security parameters, two successful polynomials were enough to recover sufficient information on the secret key.

5.4 Protection against This Attack

Hoffstein and Silverman described in [8] a similar attack but did not take into account the digital envelope. However he proposed different ways of countering it:

- Change the key very often. This solution requires that one send the actual public key to the receiver before each communication. Each time, we will need to have the new public key signed with a digital certificate, proving the origin of the key. Under these conditions, there cannot be off-line communication.
- Track decryption failure. Decryption failure should occur rarely under normal circumstances. While under a ciphertext attack, this will happens quite often. One can detect an undergoing attack and change the key. The attacker has still the power of forcing someone to change its public key when he wants.
- Induce randomness. This solution consist in adding some random px^i to the message before its decryption. This can lead to produce invalid messages from goods messages when OAEP is used. It may also produce errors in our attack, but sufficient information might still be obtained.
- Coefficient distribution analysis. The number of coefficients of the polynomial $p\phi g + fm$ falling into ranges close to $q/2$ or $-q/2$ will be larger than usual when the attack takes place. So one can discover the attack by looking counting the number of coefficients in such ranges and simply not respond to inflated polynomial.

In fact, the easiest protection against this attack is to replace the padding described in [11] by the construction from [4]. This construction works in the random oracle model and provably turns any asymmetric system into a system resistant to adaptative chosen-ciphertext attacks.

6 Conclusion

The NTRU cryptosystem makes use of the independence of reduction modulo two relatively prime integers p and q. This cryptosystem have proved secure against different attacks, such as the brute force attack, the meet-in-the-middle attack and lattice based attacks. Unfortunately, the structure of the private keys f and g opens a way to the chosen-ciphertext attack that was described here, even when the padding in [11] is used; so alternative padding/hashing methods such as those described in [4] should be used to avoid the attacks described in this paper.

References

1. Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.
2. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In A. de Santis, editor, *Advances in Cryptology — EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1994.
3. D. Coppersmith and A. Shamir. Lattice attacks on NTRU. In *Advances in Cryptology — EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 52–61, 1997.
4. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer-Verlag, 1999.
5. H. Gilbert, D. Gupta, A.M. Odlyzko, and J.-J. Quisquater. Attacks on shamir's 'rsa for paranoids'. *Information Processing Letters*, 68:197–199, 1998. http://www.research.att.com/~amo/doc/recent.html.
6. Chris Hall, Ian Goldberg, and Bruce Schneier. Reaction attacks against several public-key cryptosystems. In G. Goos, J. Hartmanis, and J; van Leeuwen, editors, *ICICS'99*, volume 1726 of *Lecture Notes in Computer Science*, pages 2–12. Springer-Verlag, 1999.
7. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring based public key cryptosystem. In *ANTS'3*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer Verlag, 1998.
8. Jeffrey Hoffstein and Joseph H. Silverman. Reaction attacks against the NTRU public key cryptosystem. Technical Report 15, NTRU Cryptosystems, August 1999.
9. M. Joye and J.-J. Quisquater. On the importance of securing your bins: the garbage-man-in-the-middle attack. *4th ACM Conf. Computer Comm. Security*, pages 135–141, 1997.
10. A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with polynomial coefficients. *Math. Annalen*, 261:515–534, 1982.
11. Joseph H. Silverman. Plaintext awareness and the NTRU PKCS. Technical Report 7, NTRU Cryptosystems, July 1998.
12. Joseph H. Silverman. Estimated breaking times for NTRU lattices. Technical Report 12, NTRU Cryptosystems, March 1999.