Recovering NTRU Secret Key From Inversion Oracles

Petros Mol[1] and Moti Yung[2]

[1] University of California, San Diego, pmol@cs.ucsd.edu
[2] Google Inc., Columbia University, moti@cs.columbia.edu

Abstract. We consider the NTRU encryption scheme as lately suggested for use, and study the connection between inverting the NTRU primitive (i.e., the one-way function over the message and the blinding information which underlies the NTRU scheme) and recovering the NTRU secret key (universal breaking). We model the inverting algorithms as black-box oracles and do not take any advantage of the internal ways by which the inversion works (namely, it does not have to be done by following the standard decryption algorithm). This allows for secret key recovery directly from the output on several inversion queries even in the absence of decryption failures. Our oracles might be queried on both valid and invalid challenges e, however they are *not required* to reply (correctly) when their input is invalid. We show that key recovery can be reduced to inverting the NTRU function. The efficiency of the reduction highly depends on the specific values of the parameters. As a side-result, we connect the collisions of the NTRU function with decryption failures which helps us gain a deeper insight into the NTRU primitive.

Key Words: NTRUEncrypt, Inversion Oracles, Universal Breaking, Public-Key Cryptanalysis

1 Introduction

For every cryptosystem the connection between recovering the secret key (i.e., universally breaking the system) and inverting the underlying (one-way) encryption function is a question of fundamental importance. The classical example is the basic Rabin cryptosystem [21] where the ability to invert instances (i.e., finding modular square roots) was shown to be equivalent to the recovery of the key, i.e., factoring; (recently, [20] extended this to all factoring based cryptosystem with a single composite). For general RSA, the question whether one can factor the modulus N querying (polynomially many times) an oracle that inverts the function $f(x) = x^e \, (mod\, N)$, remains a challenging open problem for almost 30 years (some work in the opposite direction can be found in [3]). Relating secret key recovery to ciphertext inversion may be used to strengthen security claim (in case key recovery is believed to be hard), and at the same time it opens the door to chosen ciphertext attacks as was originally pointed out by Rivest regarding Rabin's scheme.

We study this connection for the NTRU Encryption scheme (NTRUEncrypt) [1] with respect to parameter sets where the secret key f has the shape $f = 1 + p * F$ for a binary polynomial F.

We note that given the state of the art, not much is known about the structure of the NTRU encryption function and the one-way properties of the basic NTRU operation, and unlike traditional public-key schemes NTRU lacks random self-reducibility which is a property often used in understanding the structure. Our investigation, in turn, is aimed at better understanding the one-way trapdoor function that underlies NTRU.

Our conceptual goal has been a "black box" reduction, i.e., treating the inversion oracle (device) as unknown (which is a stronger reduction than ones that assume specific knowledge of how the inverting algorithm works). With this goal in mind, we found that the problem of finding the secret key pair (i.e. universally breaking the scheme) can be reformulated in a way that resembles the problem of inverting a certain instance of NTRU. More specifically, rewriting the key generation equation leaks a polynomial which, for specific parameter values, can be efficiently transformed into a valid instance and thus be recovered using a black box (hypothetical) inverting algorithm.

Related Work: To the best of our knowledge, our work is the first one that studies the problem of NTRU universal breaking outside the CCA framework. All previous key recovery attacks assume access to the decryption oracle, which on input a (valid or invalid) ciphertext applies *the standard NTRU decryption process*, and use its output to retrieve information about the secret key f. All the known CCAs are not guaranteed to work unless the decryption process functions in a very specific way. These attacks retrieve f indirectly and almost all of them work only in the presence of decryption failures.

Jaulmes and Joux [15] were the first to present CCAs against NTRU. Even though their attacks need just a small number of queries to recover f, they do not seem to work for all instantiations of NTRU and require the whole output of the decryption oracle for the recovery of f. In addition, they use *invalid* ciphertexts of a very special shape and can thus be easily thwarted by a decryption machine (which simply refuses to give an output when the input is an invalid ciphertext).

In [14] the authors present 3 new chosen-ciphertext attacks against optimized NTRU (where $f = 1+p*F$). The attacks require a very small number of queries to the decryption oracle while all the queries are on ciphertexts chosen offline and independently of the previous outputs. The main drawback of the attacks is that the oracle is queried again on *invalid ciphertexts*. In addition, the attacker needs to see the whole output of the oracle in order to fully recover the secret key f. The *reaction attacks* presented in [10] work for f of any shape and do not need to view the output of the decryption in order to recover f. The knowledge of whether the ciphertext decrypts correctly under the assumed decryption process suffices for this type of attack. The number of queries to the decryption oracle is, naturally, significantly larger than in [14].

In [12], the authors present attacks exclusively based on *valid* ciphertexts. The attacker creates the ciphertexts by encrypting valid messages and checks whether the receiver is able to decrypt them correctly (the output of the decryption is not required). These attacks work for any padding scheme and instantiation of NTRU as long as there are decryption failures. Here again the number of queries gets considerably large. In addition, these attacks seem to not have been fully implemented.

Recently, Gama and Nguyen [5] presented new CCAs on NTRU which use only valid ciphertexts chosen at random. Their attacks require the collection of a small number of decryption failures in order to recover f (but still a large number of tries in order to collect these failures). However, they require the full output of the oracle (and not just a YES/NO answer) and work only in the presence of decryption failures.

Table 1 summarizes the most representative CCAs against NTRUEncrypt. It worths noting that almost all of them (with the exception of [15] and [14]) do not work for the latest NTRU instantiations where no decryption failures occur.

Table 1. Known Chosen-Ciphertext Attacks against NTRU

Attack	**# Queries**	**Dec.Failures**	**ciphertexts**	**type of reply**	**Applicability**	**shape of F**	**Ref.**
Jaulmes, Joux	small	-	invalid	full output	unpadded version	NTRU-1998	[15]
Hong et al.	very small	-	invalid	full output	unpadded version	$1 + p * F$	[14]
Hoffstein,Silverman	large	required	invalid	YES/NO	unpadded version	any shape	[10]
How.-Graham et al.	large	required	valid	YES/NO	padded version	any shape	[12]
Gama, Nguyen	small	required	valid	full output	padded version	any shape	[5]

Our Results: All the aforementioned attacks work in the CCA framework and in particular assume access to *the decryption* oracle, while we assume access to *an inversion* oracle. Although the two approaches are not directly comparable, we present two main points that differentiate our analysis from the previous works.
(i) *We do not consider padding schemes:* After [15], several padding schemes have been proposed in order to enhance the security of NTRUEncrypt (semantic and CCA security) in the random oracle model [2] (see for example [9], [16] and several flaws pinpointed in [19] and [12]). However, here we are concerned only in the connection between breaking the primitive (that is the NTRU "one-way" function) and universal breaking. We work on the space of polynomials rather than in the space of binary strings. Thus we are not concerned about how the strings and the polynomials are connected. It is important to note that even the "valid" spaces might differ. Valid challenges e as defined below might not correspond to valid ciphertexts. Namely, there might be $e = h * r + m \,(mod\, q)$ for $(r, m) \in (\mathcal{B}(d_r), \mathcal{B})$ (valid challenge) which corresponds to an invalid ciphertext because r and m may not be connected via the hash functions used by the padding scheme. Therefore, our results do not work in the presence of a padding scheme and thus they are unlikely to lead to a practical attack. Still, the study of the unpadded version remains theoretically interesting and does say something about the NTRU primitive itself.
(ii) *The internal functionality of the oracle is not exploited:* All the aforementioned attacks assume that the oracle uses the standard decryption process (multiplication of the ciphertext e with f and then reduction modulo p). They all derive information about f *indirectly* from the effect this multiplication has on

the input of the oracle. On the contrary, here we view the inversion oracle as a black box and make no assumption on the internal computations of the oracle. This allows for key recovery even in the absence of decryption failures (NTRU-2005). Given our "lack of knowledge" about the internals of the inversion box, it is natural that we might require a relatively large number of oracle queries. Indeed, the efficiency of the reduction highly depends on the Hamming weights d_F, d_r of polynomials F and r respectively. In particular, the number of queries required to recover the secret key is exponential to $|d_F - d_r|$.
Organization: In section 2 we give some notation and a brief description of NTRUEncrypt. Section 3 defines formally the underlying NTRU primitive and studies the connection between the number of collision pairs and decryption failures. In section 4 we define the inversion oracle and its decision counterpart. Subsequently, in section 5, we give the main results and analyze the number of queries and the success probability for finding the secret key pair with respect to each oracle. Finally in section 6 we present the conclusions and suggests directions for future research.

2 NTRU Preliminaries

2.1 Definitions and Notation

We will use $\mathcal{B}$ to denote the set of all polynomials with binary coefficients. Accordingly, we use $\mathcal{B}(d)$ to indicate the set of all polynomials with exactly d 1's and all the other coefficients set to 0 (d is the hamming weight of the binary polynomial). $\mathcal{T}$ will denote the set of ternary polynomials and $\mathcal{T}(d_1, d_2)$ the set of polynomials with exactly d_1 1s and d_2 -1s. We also use the equivalence in representation between polynomials and vectors. That is, each polynomial $p(x) = \sum_{i=0}^{k} p_i x^i$ of degree k corresponds to a vector $\vec{p} = [p_0, p_1, ..., p_k]$ and vice versa. We define the *width* of a polynomial p as

$$width(p) = max(p_0, ..., p_k) - min(p_0, ..., p_k).$$

NTRU was proposed in 1996 by Hoffstein, Pipher and Silverman [8]. All the operations take place in the ring of truncated polynomials $\mathcal{P} = \mathbb{Z}_q[X]/(X^N - 1)$. That is all the polynomials involved are of degree at most $N-1$ with coefficients lying in an interval of width q. In this ring, addition of two polynomials (denoted "+") is defined as pairwise addition of the coefficients of the same degree and multiplication (denoted "*") is defined as convolution multiplication. That is

$$f(x) * g(x) = h(x) \text{ where } h_k = \sum_{i+j \equiv k \, (mod\, N)} f_i \cdot g_j \; (mod\, q).$$

The operator "*" is both commutative and associative. We define the *pseudo-inverse* of a polynomial p as the polynomial $P \in \mathcal{P}$ such that

$$P * p * s \equiv s \; (mod\, q)$$

for any polynomial $s \in \mathcal{P}$ such that $s(1) \equiv 0 \; (mod\, q)$.

2.2 Overview of NTRUEncrypt

Below we describe in brief the NTRU Encryption Scheme. Further details can be found in [8].

Parameter Set For key generation, encryption and decryption process the following parameters are used:
$-N$: Determines the maximum degree of the polynomials used. N is taken to be a prime in order to prevent attacks described by Gentry [6] and sufficiently large to prevent lattice attacks such as those described in [4] and [18]. The associated NTRU lattice seems to have dimension $2N$.
$-q$: Large modulus. It is a positive integer. Its value depends on the specific instantiation.
$-p$: Small modulus. A small integer or a polynomial with small coefficients.
N, q and p depend on the desired security level. However $(p, q) = 1$ should always hold, that is p, q should generate the unit ideal.
$-\mathcal{L}_f, \mathcal{L}_g$: Private Key spaces. Sets of polynomials from which the private keys are selected.
$-\mathcal{L}_m$: Plaintext Space. Set of polynomials that represent encoded messages.
$-\mathcal{L}_r$: Blinding value space. Set of polynomials from which the temporary blinding value used during encryption is selected.
$-\psi$: A bijection between $\mathcal{L}_m\,(mod\,p)$ and $\mathcal{L}_m$.
$-$*center:* Centering method. An algorithm that“ensures” that the reduction modulo q is performed correctly during decryption.

Key generation

Input: A prime N, the moduli p, q and a description of the sets $\mathcal{L}_f, \mathcal{L}_g$.
Output: The key pair $(pk, sk) = (h, (f, f_p))$.
1. Choose uniformly at random polynomials $f \in \mathcal{L}_f$ and $g \in \mathcal{L}_g$.
2. Compute $f_q \equiv f^{-1}\,(mod\,q)$ and $f_p \equiv f^{-1}\,(mod\,p)$. If f_q or f_p does not exist, go to previous step.
3. Compute $h \equiv f_q * p * g\,(mod\,q)$.
4. Return $(pk, sk) = (h, (f, f_p))$. $\boldsymbol{h}$ is the public key. The pair $\boldsymbol{(f, f_p)}$ is the private key.

Encryption

Input: A message $m \in \mathcal{L}_m$ and the public key h.
Output: A ciphertext e that corresponds to m.
1. Select uniformly at random a polynomial $r \in \mathcal{L}_r$ (blinding value).
2. return $e = (h * r + m)\,(mod\,q)$.

Decryption

Input: A ciphertext e and the private key pair (f, f_p).
Output: The message $m \in \mathcal{L}_m$ that corresponds to the ciphertext e.
1. Compute $a \equiv e * f \ (mod\, q)$. $(a \equiv r * h * f + f * m \equiv p * r * g + f * m \ (mod\, q))$.
2. Using a and an appropriate centering algorithm find a polynomial A such that $A = p * r * g + f * m$ in $\mathbb{Z}$ and not only $mod\, q$.
3. Compute $m \ (mod\, p) = f_p * A \ (mod\, p)$.
4. Return $\psi(m \ mod\, p) \in \mathcal{L}_m$ which corresponds to the plaintext polynomial.

Remark 2.1. In most of the instantiations of the parameter set ([1], [13]), g is also taken to be invertible $mod\, q$. In that case h is invertible too. In any case, h is pseudo-invertible $mod\, q$ with H being its pseudo-inverse.

Remark 2.2. As we mentioned in the introduction, in our analysis we do not consider padding schemes. Therefore, in the encryption and decryption process, we omit the parts that describe how padding is performed. For the padded version of encryption and decryption algorithms the reader is referred to [16], [1] and [13].

2.3 Instantiations of NTRU

Since its first publication, several variants of NTRUEncrypt have appeared in the literature. This has made the analysis of NTRU a tricky task since different choices of parameter sets might significantly affect the security of the underlying NTRU primitive. Indeed, it is not yet known whether the proposed sets lead to equivalent (in terms of security) primitives. A study of the connection of the various instantiations and an analysis of their vulnerabilities with respect to certain types of attack, consists a very challenging direction for future research.

In table 2 we summarize the main instantiations of NTRU[3] (for further details the reader is referred to [5, Section 2]). Sometimes, for efficiency reasons, a combination of the above sets might be used. For example in NTRU-2001 q might be a prime or in NTRU-2005 $\mathcal{L}_r$ and F might belong in $\mathcal{X}(d)$ which denotes the set of (binary) polynomials of the from $b_1 + b_2 * b_3$ where b_i are very sparse binary polynomials with d 1s.

3 The NTRU "One-Way" Function

In this work we consider instantiations where $f = 1 + p * F$. In these instantiations, the NTRU function is defined as follows:

[3] Recently, in order to secure against attacks presented in [11], the NTRU parameters have been revised in [7]. The major difference is that polynomials F, g, r, m belong to the space of trinary polynomials (that is their coefficients lie in the set $\{-1, 0, 1\}$). Still, in most of the new parameter sets, f has the shape $f = 1 + p * F$ with $p = 3$. We haven't looked at reductions in these new sets, but we anticipate that similar reduction arguments apply (though the number of queries required for the reduction might grow larger since the search space grows).

Table 2. The Main NTRU Parameter Sets

Variant	q	p	$\mathcal{L}_f$	$\mathcal{L}_g$	$\mathcal{L}_m$	$\mathcal{L}_r$	F	Dec. Failures	Ref.
NTRU-1998	$2^k \in [\frac{N}{2}, N]$	3	$\mathcal{T}(d_f, d_f - 1)$	$\mathcal{T}(d_g, d_g)$	$\mathcal{T}$	$\mathcal{T}(d_r, d_r)$	-	YES	[8]
NTRU-2001	$2^k \in [\frac{N}{2}, N]$	$2 + x$	$1 + p * F$	$\mathcal{B}(d_g)$	$\mathcal{B}$	$\mathcal{B}(d_r)$	$\mathcal{B}(d_F)$	YES	[16]
NTRU-2005	prime	2	$1 + p * F$	$\mathcal{B}(d_g)$	$\mathcal{B}$	$\mathcal{B}(d_r)$	$\mathcal{B}(d_F)$	NO	[13]

Definition 3.1 (The NTRU Function).

$$\mathcal{E} : \mathcal{B}(d_r) \times \mathcal{B} \rightarrow \mathbb{Z}_q^N$$
$$(r, m) \rightarrow h * r + m \ (mod\ q)$$

The NTRU function, like the underlying functions of many other practical cryptosystems, does not have a formal proof of security in that there exists no known reduction that proves that its inversion is at least as hard as a well studied hard problem. Its security appears to be related to the hardness of some lattice problems, namely the shortest and closest vector problems (SVP, CVP). In particular, finding the secret key pair (f, g) can be reduced to finding the shortest vector in a lattice constructed by the public information (L_{CS} lattice defined in [4]) whereas inverting NTRU instances can be reduced to finding the closest lattice vector to a point. However, it is possible that both NTRU problems are easier than their lattice counterparts and thus the analogy between Finding NTRU Key/Inverting challenges and SVP/CVP might be too loose.

The underlying NTRU problem can be summarized in the following definition (first formally presented by Nguyen and Pointcheval in [19])

Definition 3.2 (The NTRU Inversion Problem). *For a given security parameter k, which specifies N, p, q as well as a random public key h and $e \equiv h * r + m \, (mod\, q)$ where $m \in \mathcal{B}$ and $r \in \mathcal{B}(d_r)$, find m. Let $\mathbf{Succ}^{ow}_{NTRU}(\mathcal{A})$ denote the success probability of any adversary $\mathcal{A}$.*

$$\mathbf{Succ}^{ow}_{NTRU}(\mathcal{A}) = Pr\left[\mathcal{A}(e, h) = m \middle| (h, sk) \leftarrow \mathcal{K}(1^k), m \in \mathcal{B}, r \in_R \mathcal{B}(d_r), e \equiv h * r + m \ (mod\ q)\right]$$

The probability is taken over all the random choices made by the key generation and the encryption algorithm (h and r) as well as over all possible $m \in \mathcal{B}$. Hence, the security of NTRUEncrypt is based on the following assumption

Definition 3.3 (The NTRU Assumption). *The NTRU Inversion Problem is asymptotically hard to solve. That is, for any polynomially bounded adversary $\mathcal{A}$, $\mathbf{Succ}^{ow}_{NTRU}(\mathcal{A})$ is negligible.*

Since we are interested in efficient reductions , apart from the number of queries, we also need to bound the output of the oracles upon being asked on a specific challenge.

Definition 3.4 (Collision-Pair). *A pair $((r_1, m_1), (r_2, m_2))$ with $(r_i, m_i) \in (\mathcal{B}(d_r), \mathcal{B})$, is a NTRU collision-pair if*

$$(r_1, m_1) \neq (r_2, m_2) \quad and \quad \mathcal{E}(r_1, m_1) = \mathcal{E}(r_2, m_2).$$

Definition 3.5. *The NTRU valid challenge space is denoted by $E_{q,h}^{d_r}$ and contains the image of all pairs $(r, m) \in (\mathcal{B}(d_r), \mathcal{B})$ under NTRU function $\mathcal{E}$. Namely,*

$$E_{q,h}^{d_r} = \{e \in \mathbb{Z}_q^N | \exists r \in \mathcal{B}(d_r), m \in \mathcal{B} : e \equiv h * r + m \, (mod\, q)\}.$$

Definition 3.6. *Let $e \in \mathbb{Z}_q^N$ be a (valid or invalid) challenge. The set $preimg(e)$ is the set of all pairs $(r, m) \in (\mathcal{L}_r, \mathcal{L}_m)$ that give e under the NTRU function. That is*

$$preimg(e) = \{x_i = (r_i, m_i) | r_i \in \mathcal{L}_r, m_i \in \mathcal{L}_m, \; h * r_i + m_i \equiv e \, (mod\, q)\}$$

Obviously $|preimg(e)| = 0$ if $e \notin E_{q,h}^{d_r}$ and $|preimg(e)| \geq 1$ otherwise. The following proposition connects the number of collisions to the decryption failure probability.

Proposition 3.1. *On input $e \in E_{q,h}^{d_r}$, the standard NTRU decryption algorithm will fail to decrypt correctly with probability at least $1 - \frac{1}{|preimg(e)|}$.*

Proof. We give an intuitive proof. A less intuitive (but more formal) proof can be found in Appendix A. On input e, the standard NTRU process returns a unique message m. But there are exactly $|preimg(e)|$ distinct m's that corresponds to that e (see appendix A why these m's are distinct). Assuming (naturally) that e has emerged from the encryption of an $(r_i, m_i) \in preimg(e)$ with probability $\frac{1}{|preimg(e)|}$ (uniformly), then the inversion algorithm recovers the correct pair with probability at most $\frac{1}{|preimg(e)|}$. We say "at most" because the decryption algorithm might fail to recover any of the $(r_i, m_i) \in preimg(e)$ (due to gap or wrap failures). $\square$

The implications are straightforward. If $e \in E_{q,h}^{d_r}$ decrypts correctly, then e has a unique preimg. For example, for NTRU-2005, where decryption failures have been eliminated, this means that each valid e has a unique preimg $(r, m) \in (\mathcal{B}(r), \mathcal{B})$. Notice that the uniqueness holds not only for m (something naturally implied by perfect decryption) but for r as well. In addition, even for NTRU-2001, where decryption failures are present, the fraction of valid e that have a unique $(r, m) \in (\mathcal{B}(r), \mathcal{B})$ preimg is at least as large as the fraction of e that decrypt correctly which is (exponentially) close to one. But even for the small fraction of e that may have more than one preimages, we can argue that the number of preimages cannot grow exponentially large, otherwise the NTRU instance can be efficiently broken. Indeed, if there is a challenge e which corresponds to an exponential number of preimages, one can mount a birthday-type attack to efficiently obtain two pairs $(r_1, m_1), (r_2, m_2)$ both of which encrypt to e. We then have

$$r_1 * h + m_1 \equiv r_2 * h + m_2 \, (mod\, q) \Rightarrow (r_1 - r_2) * h \equiv m_2 - m_1 \, (mod\, q)$$

But $r_1 - r_2$ and $m_1 - m_2$ have very small norms and can be therefore used instead of f and g to invert most of the instances (of course, now the centering

algorithm will perform reduction $mod\,q$ in an interval centered at zero since $r_1 - r_2$ and $m_1 - m_2$ have coefficients in $\{-1, 0, 1\}$). We summarize the above arguments in the following sentence which we only state as an assumption for scientific accuracy.
The Preimage Assumption: For each $e \in E_{q,h}^{d_r}$ the number of pairs $(r_i, m_i) \in (\mathcal{B}(d_r), \mathcal{B})$ such that $e \equiv h * r_i + m_i\,(mod\,q)$ is polynomially bounded.

4 Modeling an Inverting Algorithm with Inversion Oracles

We will use the word "challenge" for e (instead of"ciphertext") in order to avoid any confusion with Chosen-Ciphertext Attacks. An ideal inversion algorithm would invert any valid challenge e in polynomial time given only the public information. In the rest of this section we introduce our main inversion oracle and its decision version.

Definition 4.1 **($orc1$).** *On input $e \in \mathbb{Z}_q^N$ $orc1$ outputs the pair(s) $(r, m) \in (\mathcal{B}(d_r), \mathcal{B})$ such that $e \equiv h * r + m\,(mod\,q)$ if $e \in E_{q,h}^{d_r}$. If $e \notin E_{q,h}^{d_r}$, $orc1$ gives an undefined reply denoted by "?".*

We also consider the decision version of $orc1$.

Definition 4.2 **($orc1^{DEC}$).** *On input $e \in \mathbb{Z}_q^N$, $orc1^{DEC}$ outputs "YES" if $e \in E_{q,h}^{d_r}$ and "?" otherwise.*

Remark 4.1. Both $orc1$ and $orc1^{DEC}$, as defined above, can be used to fully distinguish valid and invalid challenges. More interestingly, $orc1$ (and $orc1^{DEC}$ with a further search similar to the one described in the proof of theorem 5.3), might recover the correct message polynomials even in cases where the standard decryption might have failed (recall that the NTRUEncrypt standard decryption process in the initial instantiations has non-zero failure probability). However, the goal here is to study how easy the key recovery problem becomes in the presence of inverting algorithms, rather than argue about properties of the algorithms themselves.

5 Universal Breaking from Inversion Oracles

We denote the problem of finding the NTRU secret key pair as $\mathcal{UB}_{NTRU}$ (Universal Breaking).

Definition 5.1. *We say that $\mathcal{UB}_{NTRU}$ is (p, orc, Q)-solvable if there exists an algorithm, polynomial in the number Q of queries, which fully recovers f with probability at least p by querying oracle orc at most Q times.*

5.1 Universal Breaking Using *orc*1

Transforming the Secret Key Equation to a Valid Inversion Instance
From the key generation process we have

$$h \equiv f_q * p * g\,(mod\,q) \Rightarrow f * h \equiv p * g\,(mod\,q) \Rightarrow h * (1 + p * F) \equiv p * g\,(mod\,q)$$
$$\Rightarrow p_q * h + p_q * h * p * F \equiv g\,(mod\,q) \Rightarrow p_q * h + h * F \equiv g\,(mod\,q).$$

from which we can either get

$$h * F - g \equiv -p_q * h(mod\,q) \Rightarrow h * F + u - g \equiv u - p_q * h(mod\,q)$$

where $u(X) = X^{N-1} + X^{N-2} + ... + 1$ or alternatively

$$p_q * h \equiv -h * F + g\,(mod\,q) \Rightarrow p_q * h + h * u \equiv h * u - h * F + g\,(mod\,q).$$

If we now define $\bar{g} = u - g$, $\bar{F} = u - F$ these two give

$$\begin{array}{c} u - p_q * h \equiv h * F + \bar{g}(mod\,q) \\ p_q * h + h * u \equiv h * \bar{F} + g\,(mod\,q) \end{array} \tag{1}$$

where $h * u = (\sum h_i, \sum h_i, ..., \sum h_i)^T$. Summarizing, let $d = min\{|d_F - d_r|, |N - d_F - d_r|\}$.
Then the problem of key recovery takes the following form

$$t \equiv h * v + w\,(mod\,q) \qquad \text{(Secret Key Equation)}$$

where

- (I) $d = |d_F - d_r|$. Then $t \equiv u - p_q * h\,(mod\,q), v = F$ and $w = u - g$.
- (II) $d = |N - d_F - d_r|$. Then $t \equiv p_q * h + h * u\,(mod\,q), v = u - F$ and $w = g$.

with $u(X) = X^{N-1} + X^{N-2} + ... + 1$ (or $\vec{u} = (1, 1, ..., 1)^T$). It is important to note that in both cases w, v are binary. By definition, *orc*1 guarantees to output the correct pair(s) only when $e \in E^{d_r}_{q,h}$, that is when the blinding polynomial r used for encryption has exactly d_r 1's. Thus, in any case, in order to construct a polynomial that is"useful" for *orc*1, we need to transform (using an efficient and invertible transformation) the known polynomial t into a polynomial that belongs to the challenge space recognized by *orc*1. The steps of this transformation depend, as we show below, on the difference $d = |d_v - d_r|$ between the hamming weights of the polynomials v and r. We highlight below the aforementioned transformation.
(I) Let us consider the first case where $d = |d_F - d_r|$.
We get the following two subcases:

(a) $d_F \geq d_r$: Then $d_F - d_r = d$. We then have

$$t \equiv h * v + w\,(mod\,q), \qquad \text{where } t \equiv u - p_q * h\,(mod\,q), v = F \text{ and } w = u - g.$$

•Suppose that $d = 0$ (Binary polynomials F and r have exactly the same hamming weight). Then we query $orc1$ on $t \in E_{q,h}^{d_r}$ and by the definition of the oracle, we expect to get $F, \bar{g}$ (and thus f, g).
•Suppose that $d = 1$ and let i be an index such that $F_i = 1$. Then $h * F + \bar{g}$, can be rewritten in the following form

$$h * F + \bar{g} = h * (F + X^i - X^i) + \bar{g},$$

Thus

$$t \equiv h*(F - X^i) + h * X^i + \bar{g}\,(mod\, q) \Rightarrow t - h * X^i \equiv h*(F - X^i) + \bar{g}\,(mod\, q).$$

But $F - X^i \in \mathcal{B}(d_r)$. Querying $orc1$ on $t - h * X^i$, we can recover $F - X^i$ and consequently F (if we know i).
•Generalizing to arbitrary $d = d_F - d_r$. Suppose that we know indices $i_1, i_2, ..., i_d$ such that $F_{i_1} = F_{i_2} = ... = F_{i_d} = 1$. Then

$$t - h*(X^{i_1} + X^{i_2} + ... + X^{i_d}) \equiv h*(F - X^{i_1} - X^{i_2} - ... - X^{i_d}) + \bar{g}\,(mod\, q).$$

where again $t - h * (X^{i_1} + X^{i_2} + ... + X^{i_d}) \in E_{q,h}^{d_r}$. If we query $orc1$ on $t - h*(X^{i_1} + X^{i_2} + ... + X^{i_d})$ we can recover $F - X^{i_1} - X^{i_2} - ... - X^{i_d}$ and consequently F.
It only remains to determine the cost of finding d indices $i_1, i_2, ..., i_d \in \{0, 1, ..., N-1\}$ such that $F_{i_1} = F_{i_2} = ... = F_{i_d} = 1$.

(b) $d_F < d_r$: Then $d = d_r - d_F$.
•Suppose that for the indices $i_1, i_2, ..., i_d$ we know that $F_{i_1} = F_{i_2} = ... = F_{i_d} = 0$. Then

$$t + h*(X^{i_1} + X^{i_2} + ... + X^{i_d}) \equiv h*(F + X^{i_1} + X^{i_2} + ... + X^{i_d}) + \bar{g}\,(mod\, q).$$

If we query $orc1$ on $t + h*(X^{i_1} + X^{i_2} + ... + X^{i_d})$ we can recover $F + X^{i_1} + X^{i_2} + ... + X^{i_d}$ and consequently F.

(II) The case where $d = |N - d_F - d_r|$ is similar to case (I). Next we study the cost of finding the correct indices $i_1, i_2, ..., i_d$ that allow the reconstruction of F.

Computing the cost of finding the correct indices We consider case (Ia). The analysis of the cases (Ib),(IIa) and (IIb) is completely similar.

The input is a polynomial c with N coefficients, M of which equal 1 (of course $M \leq N$). We need to guess d indices ($d \leq M$) $i_1, ..., i_d$ such that $c_{i_1} = ... = c_{i_d} = 1$ with the least possible number of tries. The only feedback we get is a "YES" whenever $c_{i_1} = ... = c_{i_d} = 1$ holds (and then we are done) and "NO" in all other cases. Let $\mu(N, M, d)$ denote the minimum number of guesses required in the worst case, if we follow an optimal strategy and $\bar{\mu}(N, M, d)$ the expected number of guesses.

Theorem 5.1. *(i)* $\mu(N, M, d) \leq \binom{N-M+d}{d}$.
(ii) $\bar{\mu}(N, M, d) \leq \frac{\binom{N}{d}}{\binom{M}{d}}$.

Proof. (i) We restrict our guesses to the first $N-M+d$ positions of the polynomial. Suppose that the first $N-M+d$ positions contain at most $d-1$ 1's. Then the total number of 1's in the whole vector would be at most $d-1+(M-d)=M-1$ which yields a contradiction. Thus, in the worst case, we have to try at most $\binom{N-M+d}{d}$ possible (non ordered) d-tuples.
(ii) At each step we pick a set of d indices at random from all the sets of cardinality d that have not been picked in previous guesses. Obviously this yields a smaller expected number of steps than if we just picked from all possible sets (examined or not). The number of guesses in the latter scenario follows the geometrical distribution with $p=\frac{\binom{M}{d}}{\binom{N}{d}}$. Thus the expected number of the former strategy is at most $\frac{\binom{N}{d}}{\binom{M}{d}}$. □

We note that the above bounds are rather gross estimates of the values μ and $\bar{\mu}$. The problem of minimizing the number of guesses is mainly a learning problem of independent interest.

Corollary 5.1. *$\mathcal{UB}_{NTRU}$ is $(1, orc1, \mu(N, d_F, d_F-d_r))$-solvable under the Preimage Assumption.*

Proof. Getting back to case (Ia) of our problem, we are searching for $d=d_F-d_r$ 1s in a vector with $M=d_F$ 1s in order to transform $t \equiv u-p_q*h\,(mod\ q)$ which belongs to $E_{q,h}^{d_F}$ to a $t' \in E_{q,h}^{d_r}$ and then query $orc1$ on t'. After at most $\mu(N, d_F, d_F-d_r)$ guesses the decryption oracle outputs a pair $(r,m) \in (\mathcal{B}(d_r), \mathcal{B})$. Because of the Preimage Assumption, the pairs returned upon querying the oracle on a valid challenge e are polynomially bounded. This means that the dominant factor is the number of queries addressed to $orc1$ till the correct set of indices is guessed. Then, hopefully, the r returned equals $F-X^{i_1}-X^{i_2}-...-X^{i_d}$ and so F can be reconstructed correctly. There might be an exception to that. There might be a d-tuple of indices $(i'_1,...,i'_d)$ such that $t-h*(X^{i'_1}+...+X^{i'_d}) \in E_{q,h}^{d_r}$ but $F_{i'_j}=0$ for some $j \in 1,...,d$. Fortunately, we can detect these exceptions by reconstructing F'. Then either $F' \notin \mathcal{B}(d_F)$ or $g' \notin \mathcal{B}$, where $g' \equiv p_q*(1+p*F')*h\,(mod\,q)$. The preceding analysis, however, guarantees that with at most $\mu(N, d_F, d_F-d_r)$ queries to $orc1$, we will have ended up with the correct r from which F can be reconstructed in a straightforward way. Thus, the success probability after $\mu(B, d_F, d_F-d_r)$ queries is 1. □

The same result applies to cases (Ib), (IIa) and (IIb) where d is defined properly. Hence, an upper bound for the number of the oracle queries is

$$\frac{(N-d_r)!}{d!(N-d_r-d)!}=\frac{(N-d_r)!}{d!(N-d_F)!}$$

But $\frac{(N-d_r)!}{d!(N-d_r-d)!} \leq \frac{(N-d_r)^d}{d!}$. This means that if d is a (relatively small) constant, we can solve $\mathcal{UB}_{NTRU}$ in a polynomial number of queries to $orc1$.

On the contrary, the cost of the reduction grows exponentially on d. That means that, in instantiations where $d=\omega(log^{1+\epsilon}N)$ for some positive ϵ, the reduction is no longer polynomial.

Probabilistic Analysis The following theorem bounds the number of queries to $orc1$ when the success probability of solving $\mathcal{UB}_{NTRU}$ is lower-bounded by ϵ.

Theorem 5.2. *$\mathcal{UB}_{NTRU}$ is $\left(\epsilon, orc1, \binom{N}{d_F-d_r} \cdot \left(1-(1-\epsilon)^{\frac{1}{\binom{d_F}{d_F-d_r}}}\right)\right)$-solvable.*

Proof. Consider again the game of guessing d coefficients. We have in total $T = \binom{N}{d_F-d_r}$ possible (non-ordered) d-tuples ($d = d_F - d_r$), $S = \binom{d_F}{d_F-d_r}$ of which are "winning". The probability that after Q guesses we have no winning guess is

$$\begin{aligned} Pr(fail, Q) &= \left(1-\frac{S}{T}\right) \cdot \left(1-\frac{S}{T-1}\right) \cdots \left(1-\frac{S}{T-Q+1}\right) \\ &= \prod_{i=0}^{Q-1}\left(1-\frac{S}{T-i}\right) \leq \prod_{i=0}^{Q-1} e^{-\frac{S}{T-i}}, \end{aligned}$$

where we have used that for $x \geq 0$, $1-x \leq e^{-x}$ .Thus

$$Pr(fail, Q) \leq e^{-S\cdot\sum_{i=0}^{Q-1}\frac{1}{T-i}} = e^{-S\cdot(H_T - H_{T-Q})},$$

where $H_k = \sum_{i=1}^{k}\frac{1}{k}$ is the k-th Harmonic number. Let ϵ be the success probability, that is the probability that we guess a correct d-tuple in the first Q queries to $orc1$. Then using the approximation $H_k = \ln k$ for the harmonic number , we get

$$1-\epsilon = Pr(fail, Q) \leq e^{-S\cdot(H_T-H_{T-Q})} \approx e^{-S\cdot(\ln T - \ln(T-Q))} = T^{-S}(T-Q)^S.$$

Thus

$$1-\epsilon \leq \left(1-\frac{Q}{T}\right)^S \Rightarrow Q \leq T\cdot(1-(1-\epsilon)^{\frac{1}{S}}),$$

which completes the proof. ☐

5.2 Replacing *orc*1 with its Decision Version

Let us now consider the decision version of $orc1$, $orc1^{DEC}$. The main result is summarized in Theorem 5.3. First we introduce Assumption 1 that simplifies the proof of the main result and makes the combinatorial arguments more clear. We then introduce a weaker assumption (Assumption 2) and sketch how one could recover the secret key under the latter.

Assumption 1: Let $\mathcal{T}$ denote the set of all polynomials with coefficients in $\{-1, 0, 1\}$. In addition let $(r_1, m_1), (r_2, m_2) \in (\mathcal{T}, \mathcal{B})$ with $r_1(1) = r_2(1)$ and $\mathcal{E}_{q,h}(r, m) = h * r + m \,(mod\, q)$. Then

$$\mathcal{E}_{q,h}(r_1, m_1) = \mathcal{E}_{q,h}(r_2, m_2) \Leftrightarrow (r_1, m_1) = (r_2, m_2).$$

Theorem 5.3. *$\mathcal{UB}_{NTRU}$ is $(1, orc1^{DEC}, \binom{N-d_r}{d_F-d_r} + N + d_r - d_F - 1)$-solvable under Assumption 1.*

Proof. We consider again the game of guessing d 1-coefficients where now we choose the indices $(i_1, i_2, ..., i_d)$ according to the lexicographical ordering. We first exclude the $M-d$ rightmost coefficients (coefficients that correspond to positions $N-M+d, ..., N-1$) from our search. We begin with $(0, 1, ..., d-1)$ and feed $orc1^{DEC}$ with $t - h*(1 + X + ... + X^{d-1})$. At each step (and as long as we get "NO" answers by $orc1^{DEC}$) we move the rightmost index 1 position to the right until it reaches the boundary position (position $N-M+d-1$) or another index. When that happens, we move the rightmost index that can be moved 1 position to the right and initialize all its right indices right next to it (on the right). In order to make the algorithm clear, we give an example.
Let $N = 7, M = 5, d = 3$. The boundary value is $N - M + d - 1 = 4$. Then the sequence of indices we examine is the following.
(0,1,2), (0,1,3), (0,1,4), (0,2,3), (0,2,4),(0,3,4), (1,2,3), (1,2,4), (1,3,4), (2,3,4).

Notice that the number of combinations we examine is at most $\binom{N-M+d}{d}$, that is the algorithm checks all the possible (non ordered) d-combinations of the first $N-M+d$ coefficients. According to theorem 5.1 at least one of those d-tuples will result to a "YES" answer from $orc1^{DEC}$. Suppose that $orc1^{DEC}$ responds "YES" after Q queries (of course $Q \leq \binom{N-M+d}{d}$) and let $(i_1^*, ..., i_d^*)$ be the configuration of indices for which the answer is "YES". Then we know that $t - h*(X^{i_1^*} + ... + X^{i_d^*}) \in E_{q,h}^{d_r}$. But

$$t - h*(X^{i_1^*} + ... + X^{i_d^*}) \equiv h*(F - X^{i_1^*} - ... - X^{i_d^*}) + \bar{g}\,(mod\, q).$$

We claim that $F_{i_1^*} = ... = F_{i_d^*} = 1$. Indeed, suppose that $F_{i_j^*} = 0$ for some j. Then $F - X^{i_1^*} - ... - X^{i_d^*}$ is no longer binary (it has at least one -1 coefficient) but still $\mathcal{E}(F - X^{i_1^*} - ... - X^{i_d^*}, \bar{g}) \equiv \mathcal{E}(r, m)$ for a pair $(r, m) \in (\mathcal{B}(d_r), \mathcal{B})$ (recall that $t - h*(X^{i_1^*} + ... + X^{i_d^*}) \in E_{q,h}^{d_r}$). This yields a contradiction according to our assumption. Thus with at most $\binom{N-M+d}{d}$ we find d indices that correspond to 1 coefficients in F.

It only remains to recover the rest of the coefficients of F. To do this we make a simple observation. For each configuration of indices, there exists one configuration previously examined that differs in exactly one index[4]. Indeed, if we move the leftmost index that has been moved one position to the left we get a configuration of indices that has already been examined. Since the previous configuration has yielded a "NO" answer the different index corresponds to a 0 coefficient in F. So, after at most $\binom{N-M+d}{d}$ queries we know d coefficients of F that are equal to 1 and one 0 coefficient. Let $F_k = 0$ the known 0 coefficient. We

[4] There is an exception to that. When $(i_1^*, ..., i_d^*) = (0, 1, ..., d-1)$, there is no previous configuration at all. If this is the case, we can determine the rest coefficients by simply querying $orc1^{DEC}$ on $t - h*(X^{i_1^*} + ... + X^{i_{d-1}^*} + X^i)$ for each unknown coefficient F_i. Then because of the assumption, $F_i = 1$ if and only if $t - h*(X^{i_1^*} + ... + X^{i_{d-1}^*} + X^i) \in E_{q,h}^{d_r}$.

also know that

$$t - h * (X^{i_1^*} + ... + X^{i_d^*}) \equiv h * (F - X^{i_1^*} - ... - X^{i_d^*}) + \bar{g}\,(mod\, q).$$

Thus for all other unknown coefficients

$$F_i = 1 \text{ if and only if } F - X^{i_1^*} - ... - X^{i_d^*} + X^k - X^i \in \mathcal{B}(d_r)$$

or, because of the assumption, if and only if

$$t - h * (X^{i_1^*} + ... + X^{i_d^*} - X^k + X^i) \in E_{q,h}^{d_r}.$$

So we only have to query $orc1^{DEC}$ $N-d-1$ more times to fully recover F. Now, setting $M = d_F, d = d_F - d_r$, we get that we need at most $\binom{N-d_r}{d_F-d_r} + N + d_r - d_F - 1$ queries in total to recover F, which completes the proof. □

Interestingly, a similar result holds if we relax Assumption 1 to Assumption 2.
Assumption 2: Let $\mathcal{T}$ as in Assumption 1. The number of pairs $(r_i, m_i) \in (\mathcal{T}, \mathcal{B})$ with constant value $r_i(1)$ that encrypt to the same $e \in \mathbb{Z}_q^N$ under $\mathcal{E}_{q,h}$ is polynomially bounded.

Theorem 5.4. *$\mathcal{UB}_{NTRU}$ is $(1, orc1^{DEC}, \mathcal{O}(N)\cdot\binom{N-d_r}{d_F-d_r})$-solvable under Assumption 2.*

Proof (Sketch). In the presence of (polynomially many) collisions, we just need to do an extra checking every time $orc1^{DEC}$ responds "YES" in order to see if the d-tuple of indices selected is the one that leads to the correct reconstruction of F (see details of the proof for theorem 5.3). For each checking a computational overhead of $\mathcal{O}(N)$ queries is added (the checking works in a way similar to the checking in the proof of theorem 5.3). In that case the total number of queries to $orc1^{DEC}$ is multiplied by a factor of at most $\mathcal{O}(N)$. □

Remark 5.1. The above analysis implies that if $d_F - d_r$ is small with respect to N, we can universally break NTRUEncrypt if we have a polynomial time distinguisher between valid and invalid challenges.

Decryption Oracles and Real NTRU Parameters The applicability of our reductions is enhanced by the set of parameters that have been proposed from time to time. Indeed both in [13] and in [1] it is suggested that during the key generation process, d_F is set equal to d_r. In addition, in the web challenges published by NTRU Cryptosystems (www.ntru.com/cryptolab/challenges.htm),the parameter sets proposed are as shown in the table below

Security	**N**	**q**	d_F	d_g	d_r
Medium	251	128	72	71	72
High	347	128	64	173	64
Highest	503	256	420	251	170

For the Medium and High level of security $d_r = d_F$, which, suggests that for theses values of parameters the problems of inverting a challenge e and finding the secret key pair, are structurally the same. For the highest level of security, however, $d = 420 + 170 - 503 = 87$ which does not allow for efficient reductions.

6 Conclusions

We have shown how inversion black-box oracles that output message polynomials corresponding to valid challenges e or that serve as decision oracles lead to a secret key recovery in the current NTRU system where $f = 1 + p * F$. The cost of recovering the secret key depends on the difference between the Hamming weights of the polynomials F and r in an exponential fashion. The reductions presented do not work in the presence of a padding scheme and thus seem unlikely to lead to any practical attacks. Still, this fundamental connection teaches us about the very structure of the cryptosystem in general. The implication is quite straightforward and should be carefully interpreted: Finding an algorithm that inverts NTRU instances in recent NTRU instantiations (and for certain parameter values), opens the door to secret key recovery within a small number of queries to that algorithm. It is important to note that there is nothing particular that makes the secret key recovery harder than inverting random instances (see equation Secret Key Equation). Indeed, the target challenge t is no less "random" than any other inversion instance, since F, g are random polynomials.

As a related future direction, we believe that coming up with more efficient reductions which further exploit the structure of the NTRU function is an interesting field for investigation. Finally, another challenging direction would be to extend the range of behavior of the black-box oracles to non-ideal ones (that fail with some probability to return the correct preimage even when being queried on valid challenges).

References

1. *EESS:Consortium for Efficient Embedded Security. Efficient Embedded Security Standards #1:Implementation Aspects of NTRU and NSS*, draft version 3.0 edition, July 2001.
2. Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
3. Dan Boneh and Ramarathnam Venkatesan. Breaking RSA May Not Be Equivalent to Factoring. In Kaisa Nyberg, editor, *EUROCRYPT*, volume 1403 of *Lecture Notes in Computer Science*, pages 59–71. Springer, 1998.
4. Don Coppersmith and Adi Shamir. Lattice Attacks on NTRU. In Walter Fumy, editor, *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 52–61. Springer, 1997.
5. Nicolas Gama and Phong Q. Nguyen. New Chosen-Ciphertext Attacks on NTRU. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 89–106. Springer, 2007.
6. Craig Gentry. Key Recovery and Message Attacks on NTRU-Composite. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 182–194. Springer, 2001.
7. Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. Hybrid Lattice Reduction and Meet in the Middle Resistant Parameter Selection for NTRUEncrypt. Available at grouper.ieee.org/groups/1363/lattPK/submissions/ChoosingNewParameters.pdf.
8. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A Ring-Based Public Key Cryptosystem. In Joe Buhler, editor, *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.
9. Jeffrey Hoffstein and Joseph H. Silverman. Protecting NTRU Against Chosen Ciphertext and Reaction Attacks. Technical report, NTRU Cryptosystems, citeseer.ist.psu.edu/hoffstein00protecting.html, 2000.
10. Jeffrey Hoffstein and Joseph H. Silverman. Reaction Attacks Against the NTRU Public Key Cryptosystem. Technical Report, NTRU Cryptosystems, citeseer.ist.psu.edu/hoffstein00reaction.html, June 2000. Report #015,version 2.
11. Nick Howgrave-Graham. A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 150–169. Springer, 2007.
12. Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H. Silverman, Ari Singer, and William Whyte. The Impact of Decryption Failures on the Security of NTRU Encryption. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 226–246. Springer, 2003.
13. Nick Howgrave-Graham, Joseph H. Silverman, and William Whyte. Choosing Parameter Sets for NTRUEncrypt with NAEP and SVES-3. Technical Report, NTRU CRYPTOSYSTEMS, 2005.
14. J. Hong and J. Han and D. Kwon and D. Han. Chosen-Ciphertext Attacks on Optimized NTRU. 2002. Cryptology ePrint Archive: Report 2002/188.
15. Éliane Jaulmes and Antoine Joux. A Chosen-Ciphertext Attack against NTRU. In Mihir Bellare, editor, *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2000.
16. Jeffrey Hoffstein and Joseph Silverman. Optimizations for NTRU. Technical report, NTRU Cryptosystems, citeseer.ist.psu.edu/693057.html, June 2000.

17. Mats Näslund and Igor Shparlinski and William Whyte. On the Bit Security of NTRUEncrypt. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 62–70. Springer, 2003.
18. Alexander May. Cryptanalysis of NTRU-107. Available at www.informatik.tu-darmstadt.de/KP/publications/01/CryptanalysisOfNTRU.ps, 1999.
19. Phong Q. Nguyen and David Pointcheval. Analysis and Improvements of NTRU Encryption Paddings. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 210–225. Springer, 2002.
20. Pascal Paillier and Jorge Luis Villar. Trading One-Wayness Against Chosen-Ciphertext Security in Factoring-Based Encryption. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 252–266. Springer, 2006.
21. M. O. Rabin. Digital Signatures and Public-Key Functions as Intractable as Factorization. Technical report, Cambridge, MA, USA, 1979.

A Proof of Theorem 3.1

Proof. For each pair $(r_i, m_i) \in preimg(e)$, we define $a_i = p * g * r_i + f * m_i$ where, as usual, f, g are the secret and auxiliary key respectively. Equation $e \equiv h * r_i + m_i \,(mod\; q)$ gives $f * e \equiv a_i \,(mod\; q)$. We need the following two lemmas.

Lemma A.1. *If $(r_i, m_i), (r_j, m_j)$ are two distinct pairs that belong to $preimg(e)$, then $(r_i \neq r_j) \wedge (m_i \neq m_j)$.*

Proof. Suppose on the contrary, that there exist $(r_i, m_i), (r_j, m_j)$ with $(r_i, m_i) \neq (r_j, m_j)$ such that $(r_i = r_j) \vee (m_i = m_j)$. Then we have the following two cases

(a) $r_i = r_j$: Then

$$h * r_i + m_i \equiv h * r_j + m_j \,(mod\; q) \overset{r_i = r_j}{\Rightarrow} m_i \equiv m_j \,(mod\; q).$$

But both $m_i, m_j \in \mathcal{L}_m$ and thus have small coefficients (with respect to q). Therefore $m_i = m_j$ holds over the integers which yields a contradiction.

(b) $m_1 = m_2$: Then we have

$$h * r_1 \equiv h * r_2 \,(mod\; q) \Rightarrow h * (r_1 - r_2) \equiv 0 \,(mod\; q)$$

But h has a pseudo-inverse, that is there exists a polynomial $H \in \mathcal{P}$ such that $H * h * s \equiv s \,(mod q)$ for any polynomial s with $s(1) \equiv 0 \,(mod q)$. Now notice that $(r_1 - r_2)(1) = r_1(1) - r_2(1) = d_r - d_r = 0$ (in all instantiations of NTRU the value $r(1)$ is a public constant). This gives that $H * h * (r_1 - r_2) \equiv r_1 - r_2 \,(modq)$, which combined with the above equation gives $r_1 - r_2 \equiv 0 \,(modq)$. This implies that $r_1 = r_2$ since both r_1 and r_2 have very small coefficients.

□

Lemma A.2. *$a_i \neq a_j$ over $\mathbb{Z}\, \forall\, i \neq j$. That is $a_i s$ are pairwise distinct.*

Proof. Suppose that there exist distinct indices i, j such that $a_i = a_j$. First observe that $(r_i \neq r_j) \wedge (m_i \neq m_j)$, otherwise we would have

$$p * g * r_i + f * m_i = p * g * r_j + f * m_j \overset{\times f_q}{\Rightarrow} h * r_i + m_i \equiv h * r_j + f * m_j \,(mod\; q)$$

which clearly contradicts lemma A.1. If we multiply both sides with f_p (recall that $f_p * f = 1 + p * k$ for a polynomial k) we get

$$p * f_p * g * r_i + (1 + p * k) * m_i = p * f_p * g * r_j + (1 + p * k) * m_j \quad \text{over the integers}$$

which gives $m_i \equiv m_j \,(mod p)$. But p and the modulo p reduction process are selected in such a way that $m \,(mod p)$ for a polynomial $m \in \mathcal{L}_m$ uniquely determines m. Otherwise the decryption would be ambiguous. This means that $m_i = m_j$ over the integers which gives a contradiction. □

Back to the proof of 3.1, we have that for each pair of distinct indices i, j $a_i \neq a_j$ but $a_i \equiv a_j \, (mod \, q)$ for all pairs that collide to the same e, since $a_i \equiv a_j \equiv f * e \, (mod \, q)$. This means that there exists *at most* one index i such that all the coefficients of a_i lie in the interval dictated by the centering algorithm (let's say $[A, A+q-1]$). Indeed, if again $a_i, a_j, i \neq j$ had all their coefficients in $[A, A+q-1]$ (of range q) the equation $a_i \equiv a_j \, (mod \, q)$ would imply $a_i = a_j$ over the integers (contradiction).

Thus, the centering algorithm (and the inversion part of the decryption algorithm in general) works properly for at most one pair $(r_i, m_i) \in preimg(e)$. All the decryption algorithm sees is the challenge e and has no information on the preimage pair (r, m). Assuming (naturally) that e has emerged from the encryption of each $(r_i, m_i) \in preimg(e)$ with probability $\frac{1}{|preimg(e)|}$ (uniformly), with probability at most $\frac{1}{|preimg(e)|}$ the inversion algorithm recovers the correct pair. Thus we conclude that

$$Pr[Decryption \, succeeds | input \, is \, e] \leq \frac{1}{|preimg(e)|}.$$

$\square$