NTRU: A NEW HIGH SPEED PUBLIC KEY CRYPTOSYSTEM

JEFFREY HOFFSTEIN, JILL PIPHER, JOSEPH H. SILVERMAN

Brown University

August 13, 1996

Preliminary Draft

ABSTRACT. We describe NTRU, a new public key cryptosystem. NTRU features short, easily created keys, high speed, and low memory requirements. NTRU encoding and decoding uses a mixing system suggested by polynomial algebra combined with a clustering principle based on elementary probability theory. The security of the NTRU cryptosystem comes from the interaction of the polynomial mixing system with the independence of reduction modulo two relatively prime integers p and q.

CONTENTS

Key words and phrases. coding theory, public key cryptosystem, NTRU cryptosystem.

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX

0. Introduction

There has been considerable interest in the creation of efficient and computationally inexpensive public key cryptosystems since Diffie and Hellman [3] explained how such systems could be created using one-way functions. Currently, the most widely used public key system is RSA, which was created by Rivest, Shamir and Adelman in 1978 [7].

In this paper we describe a new public key cryptosystem, which we call the NTRU system. The encoding procedure uses a mixing system based on polynomial algebra and reduction modulo two numbers p and q, while the decoding procedure uses an unmixing system whose validity depends on elementary probability theory. The security of the NTRU public key cryptosystem comes from the interaction of the polynomial mixing system with the independence of reduction modulo p and q.

Encoding and decoding with NTRU are extremely fast, and key creation is fast and easy. See section 1.5 for specifics, but we note here that NTRU takes $O(N^2)$ operations to encode or decode a message block of length N, making it considerably faster than the $O(N^3)$ operations required by RSA.

Acknowledgements. We would like to thank Hendrik Lenstra Jr., Bjorn Poonen, and Benne de Weger for their help with lattice reduction methods, Andrew Odlyzko for pointing out the meet-in-the-middle attack and other helpful suggestions, Mike Rosen for his help with polynomial inverses, and Dan Lieman for his assistance in all phases of this project.

1. Description of the NTRU algorithm

In this section we give a description of the NTRU public key cryptosystem, a high speed, short key, low memory, public key cryptosystem based on polynomial multiplication and residues. NTRU admits many variants which take into account the relative computing power of the encoder and decoder and the relative importance of speed versus the probability that an occasional message will be undecipherable. This section gives one version of NTRU; for a discussion of some possible variations and generalizations, see section 4.

We also mention that we will constantly refer to polynomial multiplications and divisions with remainder, but most of these are of a very special sort which allow for extremely fast computations. (See section 7 for a sample implementation.)

1.1. Notation and parameters. An NTRU cryptosystem depends on three integer parameters (N,p,q) and a set of polynomials $\mathcal{L}$ of degree N with integer coefficients. For example, $\mathcal{L}$ might equal

$$\mathcal{L} = \left\{ \sum_{k=0}^{N-1} \varepsilon_k x^k : \begin{array}{c} \text{exactly } d \text{ of the } \varepsilon_k\text{'s equal one} \\ \text{and the remaining } \varepsilon_k\text{'s equal zero} \end{array} \right\},$$

in which case we will speak of *(N,p,q,d)-binary NTRU.* As another example, the coefficients of the polynomials in $\mathcal{L}$ might be taken from a larger set, such as

$$\mathcal{L} = \left\{ \sum_{k=0}^{N-1} \varepsilon_k x^k : \begin{array}{c} \text{exactly } d \text{ of the } \varepsilon_k\text{'s equal to each of } -r, -r+1, \dots, -2, -1 \\ \text{exactly } d \text{ of the } \varepsilon_k\text{'s equal to each of } 2, 3, \dots, r \\ \text{exactly } d+1 \text{ of the } \varepsilon_k\text{'s equal } 1, \\ \text{the remaining } N - rd - 1 \text{ of the } \varepsilon_k\text{'s equal } 0 \end{array} \right\}.$$

We will call this choice of $\mathcal{L}$ an (N,p,q,d,r)-*symmetric NTRU*. (Of course, it's not quite symmetric, due to the extra 1; we'll see in section 1.6 why the extra 1 is needed.) See section 1.5 for some sample parameter choices and associated operating specifications, and section 5 for a general discussion of parameter selection.

A basic NTRU operation is multiplication of polynomials modulo $x^N - 1$. We will generally identify a polynomial with the vector of its coefficients,

$$F = \sum_{i=1}^{N} F_i x^{N-i} = [F_1, F_2, \dots, F_N].$$

We write $\circledast$ to denote multiplication in the quotient ring $\mathbb{Z}[x]/(x^N - 1)$. This *star multiplication* is given explicitly by the simple formula

$$F \circledast G = H \qquad \text{with} \qquad H_k = \sum_{j=1}^{k-1} F_j G_{k-j} + \sum_{j=k}^{N} F_j G_{N+k-j}.$$

We mention that a $\circledast$-product in which one of the polynomials is a binary polynomial (i.e., polynomials whose coefficients are all 0's and 1's) can be done without multiplications. (See the pseudo-code implementation of NTRU in section 7.)

1.2 Public and private keys. The parameters (N,q,p) and the set $\mathcal{L}$ are public knowledge. An NTRU public key/private key pair consists of two polynomials of the following form:

Private key: $f = [f_1, f_2, \dots, f_N]$ with $f \in \mathcal{L}$.
Public key: $h = [h_1, h_2, \dots, h_N]$ with $0 \le h_i < q$.

The private key may be chosen (almost) randomly from the $\#\mathcal{L}$ possibilities. We will explain below (section 1.6) how to use the private key f to create the public key h.

1.3 Encoding a message. Suppose that Cathy (the encoder) wants to send a message to Dan (the decoder) using Dan's public key h and parameters (N,q,p) and set $\mathcal{L}$. Cathy first breaks her message into message blocks of the form

$$m = [m_1, m_2, \dots, m_N] \qquad \text{with } 0 \le m_i < p.$$

In order to encode a message block, Cathy chooses a random polynomial from $\mathcal{L}$ (which we call the *encoding fuzz*)

$$\phi = [\phi_1, \phi_2, \dots, \phi_N] \in \mathcal{L}.$$

Cathy then computes

$$e = \phi \circledast h + m \pmod{q}. \tag{1}$$

This is the encoded message which Cathy transmits to Dan.

1.4 Decoding a message. Suppose that Dan has received the message e from Cathy and wants to decode it using his private key f. To do this efficiently, Dan should have precomputed two numbers s and t and a certain polynomial

$$f_p^{-1} = [(f_p^{-1})_1, (f_p^{-1})_2, \dots, (f_p^{-1})_N] \qquad \text{satisfying } 0 \le (f_p^{-1})_i < p.$$

We will defer giving the precise definitions of s, t and f_p^{-1} until section 1.6 below.

In order to decode e, Dan first computes

$$a = f \circledast e \pmod{q}.$$

Next he creates a shifted polynomial according to the rule

$$b = [b_1, b_2, \dots, b_N] \qquad \text{with} \quad b_k = a_k + t - \begin{cases} 0 & \text{if } a_k < s, \\ q & \text{if } a_k \ge s. \end{cases}$$

Finally, Dan recovers the original message by computing

$$b \circledast f_p^{-1} \pmod{p}.$$

(N.B. The congruence modulus has "mysteriously" switched from q to p. This interaction of p and q, combined with the special form of the private key f, is what makes NTRU secure.)

Remark. For appropriate parameter values, there is an extremely high probability that the decoding procedure will recover the original message. For example, it is not difficult to create NTRU cryptosystems with a decoding failure rate of 1 message in 10^{10} (or even better), see section 3. However, some parameter choices may cause occasional decoding failure, so one should probably include a few check bits in each message block. The usual cause of decoding failure will be that the precomputed s value is not correct, in which case the message can usually be recovered by using successively $s-1$, $s+1$, $s-2$, $s+2$, ... in place of s. If no value of s works, then we say that we have *gap failure* and the message cannot be decoded. For well-chosen parameter values, this will occur so rarely that it can be ignored in practice.

1.5 Encoding and decoding speed and memory requirements. Computation of an arbitrary product $F \circledast G$ requires N^2 multiplications and N^2 additions. If we also want to reduce the coefficients modulo p or q, this requires an additional N divisions with remainder. Thus it takes $O(N^2)$ operations to encode a message block m of length $N \log_2 p$ bits, the encoded message has length $N \log_2 q$ bits, and it takes $O(N^2)$ operations to decode the message block. In practice, encoding also requires the generation of some random bits (i.e., to choose a random ϕ from $\mathcal{L}$), but this is compensated for by the fact that encoding only requires one $\circledast$-multiplication, while decoding requires two. In any case, both encoding and decoding take $O(N^2)$ operations for messages of length $O(N)$-bits. This may be compared with RSA, which requires $O(N^3)$ operations to encode/decode (although the use of small exponents can reduce RSA encoding to around $O(N^2)$). Finally we note that the number of available keys is (approximately) $\#\mathcal{L}$, which gives a security level of $(\#\mathcal{L})^{1/2}$ due to the existence of a standard meet-in-the-middle attack (see section 6.2).

Remark. In principle, the private key consists merely of the polynomial f, since one can always use f to compute f_p^{-1}. However, in practice Dan would not want to recompute f_p^{-1} each time he receives a message, so for most applications the storage required for the private key is the length of f and f_p^{-1} together.

Tables 1 and 2 illustrate key size, security, message expansion, and encoding/decoding speed for various parameter choices. The first table deals with binary (N,p,q,d)-NTRU, which means that $\mathcal{L}$ contains all polynomials whose coefficients consist of d ones and $N-d$ zeros. The second table describes symmetric (N,p,q,d,r)-NTRU, where the polynomials in $\mathcal{L}$ have exactly d coefficients equal to each value between $-r$ and r other than 0 and 1, plus $d+1$ coefficients equal to 1 and the remaining $N-2d$ coefficients equal to 0. In all cases we have chosen parameters so the the probability of decoding failure is very small. See section 3 for a rigorous statistical analysis of one choice of parameters. We also mention that the "operations" for binary NTRU are mostly additions, while the symmetric NTRU operations are multiplications of the form $u\cdot v$ with $u\approx p$ and $v\approx q$; so although symmetric NTRU will usually be faster than binary NTRU, the speed difference may not be quite as much as the tables seem to indicate. In any case, NTRU should be between one and two orders of magnitude faster than RSA at similar security levels.

1.6. Creation of encoding and decoding keys. In order to create his public and private keys, Dan chooses two random polynomials

$$f=[f_1,f_2,\dots,f_N] \qquad \text{and} \qquad g=[g_1,g_2,\dots,g_N]$$

from the set $\mathcal{L}$. For example, if he is using binary (N,p,q,d) NTRU, then f and g would be randomly chosen polynomials of degree $N-1$ whose coefficients consist of d ones and $N-d$ zeros. The polynomial f is the *private decoding key* and should be kept secret. The polynomial g will be used to create the public key, but after that it will not be needed again and may be discarded. The private key f is required to have two further properties:

$$f \text{ has an inverse modulo the ideal } (q,x^N-1).$$
$$f \text{ has an inverse modulo the ideal } (p,x^N-1).$$

(Aside. The parameters must be chosen to ensure that $\gcd(f(1),pq)=1$, since otherwise $f(x)$ will be divisible by $x-1$ modulo some prime divisor of pq. This is why there's the "extra" 1 coefficient in symmetric NTRU, since without it we would have $f(1)=0$.) For our suggested parameter values (Tables 1 and 2), virtually every f will have these properties. (See section 5.) We will denote the two inverses of f by f_q^{-1} and f_p^{-1} respectively. In other words,

$$f_q^{-1}=[w_1,w_2,\dots,w_N] \quad \text{with } 0\le w_i<q \text{ and } \quad f\circledast f_q^{-1}\equiv 1 \pmod q. \qquad (2)$$
$$f_p^{-1}=[v_1,v_2,\dots,v_N] \quad \text{with } 0\le v_i<p \text{ and } \quad f\circledast f_p^{-1}\equiv 1 \pmod p. \qquad (3)$$

There are fast algorithms for computing f_q^{-1} and f_p^{-1}, see section 7. Further, f_q^{-1} and f_p^{-1} need only be computed once for each code. The polynomial f_q^{-1} will be

	N / p / q / d	107 / 5 / 256 / 31	167 / 7 / 512 / 71	167 / 15 / 1024 / 71
security	$\binom{N}{d}^{1/2}$	2^{45}	2^{80}	2^{80}
public key (bits)	$N\lceil\log_2 q\rceil$	856	1503	1670
private key (bits)	$N\lceil\log_2 2p\rceil$	428	668	835
expansion	$\log q/\log p$	3.45	3.21	2.56
message block (bits)	$N\log_2 p$	248	469	652
encoding (per 512 bits): random bits	$\approx 512(d/N)\log_p N$	≈ 431	≈ 573	≈ 411
operations	$\approx 1024N/\log_2 p$	$\approx 47K$	$\approx 61K$	$\approx 44K$
decoding (per 512 bits): operations	$\approx 2048N/\log_2 p$	$\approx 94K$	$\approx 122K$	$\approx 88K$

Table 1. Binary (N,p,q,d) NTRU

	N / p / q / d / r	107 / 7 / 512 / 31 / 1	83 / 5 / 512 / 11 / 2	83 / 13 / 1024 / 11 / 2
security	$C(N,d,r)^{1/2}$†	2^{80}	2^{80}	2^{80}
public key (bits)	$N\lceil\log_2 q\rceil$	963	747	830
private key (bits)	$N\lceil\log_2 2rp\rceil$	428	415	498
expansion	$\log q/\log p$	3.21	3.88	2.70
message block (bits)	$N\log_2 p$	300	193	307
encoding (per 512 bits): random bits	$\approx 1024(dr/N)\log_p N$	≈ 712	≈ 745	≈ 467
operations	$\approx 1024N/\log_2 p$	$\approx 39K$	$\approx 37K$	$\approx 23K$
decoding (per 512 bits): operations	$\approx 2048N/\log_2 p$	$\approx 78K$	$\approx 73K$	$\approx 46K$

† $C(N,d,r)=N!/((N-2dr-1)!\cdot d!^{2r-1}(d+1)!)$

Table 2. Symmetric (N,p,q,d,r) NTRU

used to create the encoding key, but after that it may be discarded. The polynomial f_p^{-1} is used in the decoding process. It should be precomputed and saved.

In order to create his public key, Dan multiplies $f \circledast g$ and uses the product to compute a *parity polynomial*

$$\pi = [\pi_1, \pi_2, \dots, \pi_N] \qquad \text{with} \quad \pi + f \circledast g \equiv 0 \pmod{p} \quad \text{and} \quad 0 \le \pi_i < p. \tag{4}$$

In other words, the π_k's are chosen so that every coefficient of $\pi + f \circledast g$ is divisible by p. Then Dan's *public encoding key h* is the polynomial

$$h = [h_1, h_2, \dots, h_N] = \pi \circledast f_q^{-1} + g \pmod{q}. \tag{5}$$

Notice that $0 \le h_k < q$, so the public key consists of N numbers in the range 0 to $q-1$.

Finally, Dan needs to precompute the two shift values s and t. These depend on the set $\mathcal{L}$, or more precisely, on the average value of the coefficients of the polynomials in $\mathcal{L}$. For binary (N, p, q, d) NTRU, Dan's s and t are

$$s \equiv \left\lfloor \frac{q}{2} + d(p-1) + \frac{d^3}{N} \right\rceil \pmod{q}, \qquad \text{with } 0 \le s < q.$$
$$t \equiv q \left\lfloor \frac{d(p-1) + d^3/N}{q} \right\rceil \pmod{p}, \qquad \text{with } 0 \le t < p.$$

For symmetric (N, p, q, d, r) NTRU, his s and t are simply

$$s = \left\lfloor \frac{q}{2} + p - 1 \right\rceil \qquad \text{and} \qquad t = 0.$$

2 Why NTRU works

In this section we explain why the decoding process retrieves the original message. We suppose that the plaintext message m has been encoded using the encoding key h, yielding the encoded message e. Thus

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

To decode, we compute $e \circledast f \pmod{q}$.

Recall that h was chosen to satisfy

$$h \equiv \pi \circledast f_q^{-1} + g \pmod{q}.$$

Hence

$$\begin{aligned}
e \circledast f &\equiv (\phi \circledast h + m) \circledast f \pmod{q} && \text{from (1)} \\
&\equiv (\phi \circledast (\pi \circledast f_q^{-1} + g) + m) \circledast f \pmod{q} && \text{from (5)} \\
&\equiv \phi \circledast (\pi \circledast f_q^{-1} \circledast f + g \circledast f) + m \circledast f \pmod{q} && \\
&\equiv \phi \circledast (\pi + g \circledast f) + m \circledast f \pmod{q} && \text{from (2).}
\end{aligned}$$

This computation has been done modulo q, which means that the coefficients are reduced to lie in the range 0 to $q-1$. In other words, the message e and decoding key f allow us to determine coefficients a_k satisfying

$$e \circledast f \equiv [a_1, a_2, \ldots, a_N] \pmod{q} \qquad \text{with } 0 \le a_k < q.$$

Further, we know from (4) that all of the coefficients of

$$\pi + g \circledast f$$

are divisible by p. Unfortunately, in the above calculation we have reduced modulo q, which would seem to destroy all mod p information.

Suppose that we write

$$\phi \circledast (\pi + f \circledast g) + m \circledast f \equiv [A_1, A_2, \ldots, A_N].$$

N.B. We are not reducing the coefficients modulo q, and we do not yet know the value of the A_k's. However, we do know the following two facts:

(i) $A_k \equiv a_k \pmod{q}$, where $0 \le a_k < q$ and the a_k's are known.
(ii) With high probability, the interval from the smallest A_k to the largest A_k is considerably smaller than q, and the A_k's tend to cluster around the middle of this interval.

(See section 3 for a justification of (ii).) We now explain how the decoding process enables us to compute the value of the A_k's modulo p. For concreteness, we will treat the case of binary (N,p,q,d) NTRU. We leave it to the reader to make the necessary minor changes for symmetric (N,p,q,d,r) NTRU or other variations.

We begin by computing the expected magnitude of the A_k's. The polynomials f and g each have d ones as coefficients, so an average coefficient of $f \circledast g$ has size d^2/N. (Intuitively, $f \circledast g$ says to choose d "random" coefficients of g and add them up, and a random coefficient of g has a d/N chance of being a one.) Similarly, an average coefficient of $\phi \circledast f \circledast g$ has size d^3/N, an average coefficient of $\phi \circledast \pi$ has size $d(p-1)/2$, and an average coefficient of $f \circledast m$ has size $d(p-1)/2$. These last two values follow from the fact that on average, the coefficients of π and m will consist of an equal number of 0's, 1's, ... , $(p-1)$'s. Adding these values together, we see that

$$\text{Average coefficient of } \phi \circledast (\pi + f \circledast g) + m \circledast f \text{ has size } d(p-1) + \frac{d^3}{N}.$$

(For symmetric NTRU, this value is $p - 1 + 1/N$.)

Consider now the interval from the smallest A_k to the largest A_k. We know that the center of this interval will be approximately $d(p-1) + d^3/N$, so if we let

$$Q = \left\lfloor \frac{d(p-1) + d^3/N}{q} \right\rceil, \tag{6}$$

then qQ will be the multiple of q lying in (or closest to) the interval. (For symmetric NTRU, $Q = 0$, since we would always take $p \ll q$.) We will suppose that qQ actually lies amongst the A_k's. (This is the hardest case to analyze; we will leave the other cases to the reader.) Choose $R_1, R_2 \ge 0$ so that all of the A_k's lie in the interval from $qQ - R_1$ to $qQ + R_2$, as illustrated in Figure 1.

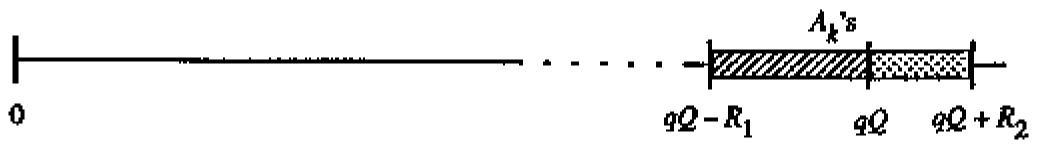

Figure 1

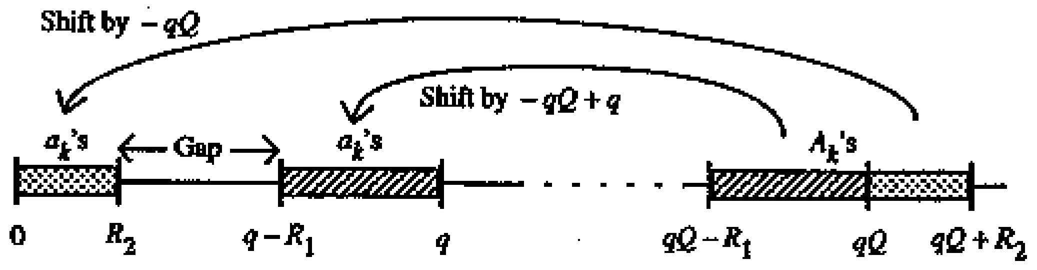

Figure 2

We observe that the center of this interval will generally be very close to the average magnitude of the A_k's. In other words,

$$\frac{(qQ+R_2)+(qQ-R_1)}{2}=\frac{R_2-R_1}{2}+qQ\approx d(p-1)+\frac{d^3}{N}. \tag{7}$$

Notice that the a_k's, which we know, are obtained by shifting the A_k's modulo q into the interval from 0 to $q-1$. Referring to Figure 2, we see that some of the A_k's are shifted by $-qQ$ into the interval from 0 to R_2, and the others are shifted by $-qQ+q$ into the interval from $q-R_1$ to $q-1$. More precisely,

$$a_k=\begin{cases} A_k-qQ & \text{if } A_k\ge qQ, \\ A_k-qQ+q & \text{if } A_k<qQ. \end{cases}$$

Thus knowing the a_k's does not directly allow us to determine the value of the A_k's modulo p.

However, this reduction process leaves a *gap* in the a_k's running from R_2 to $q-R_1$. The midpoint of this gap is $(q-R_1+R_2)/2$, so using (7), we find that the midpoint of the gap is approximately

$$\frac{q}{2}+d(p-1)+\frac{d^3}{N}-qQ.$$

If we set

$$s\equiv\left\lfloor\frac{q}{2}+d(p-1)+\frac{d^3}{N}\right\rceil \pmod q \qquad \text{with } 0\le s<q,$$

then there is a very high probability that s will lie in the gap. A nice feature of NTRU decoding is that it works as long as s lies somewhere in the gap; it doesn't make any difference whether it lies in the middle or near the edge.

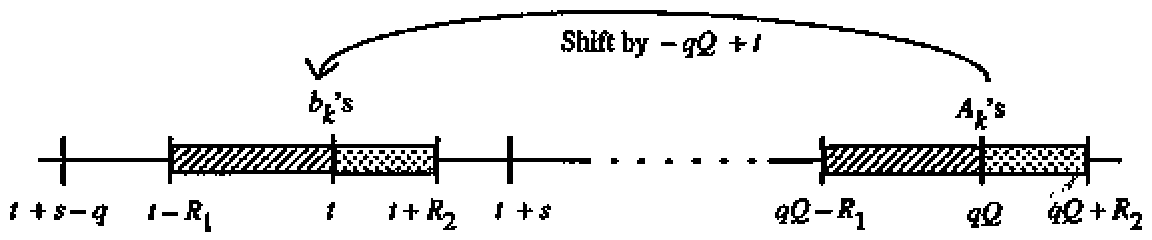

Figure 3

The number s lies between the lefthand clump of a_k's and the righthand clump of a_k's, so if we choose the b_k's as described in section 1.4, we find that

$$b_k = \begin{cases} a_k + t = A_k - qQ + t & \text{if } a_k < s, \\ a_k + t - q = A_k - qQ + t & \text{if } a_k \geq s. \end{cases}$$

Thus the b_k's are related to the A_k's by the rule

$$b_k = A_k - qQ + t \qquad \text{for all } 0 \leq k < N, \tag{8}$$

where t has been chosen to satisfy $t \equiv qQ \pmod{p}$. (For symmetric NTRU, we take $t = 0$.) See Figure 3.

There are two fundamental observations to make here. First, the formula (8) is an exact equality of integers, and second, the quantities qQ and t are fixed, independent of k. Now the b_k's are known quantities, so we can take the formula $b_k = A_k - qQ + t$ and reduce it modulo p to compute

$$\begin{aligned} b &= [b_1, b_2, \ldots, b_N] \\ &\equiv [A_1 - qQ + t, A_2 - qQ + t, \ldots, A_N - qQ + t] \pmod{p} \\ &\equiv [A_1, A_2, \ldots, A_N] \qquad \text{since } t \equiv qQ \pmod{p}, \\ &\equiv \phi \circledast (\pi + f \circledast g) + m \circledast f \pmod{p}. \end{aligned}$$

Note that the coefficients are now being reduced modulo p, not modulo q. We know from (4) that

$$\pi + g \circledast f \equiv 0 \pmod{p},$$

so starting from the encoded message e and the private key f, we have recovered the value of

$$b \equiv m \circledast f \pmod{p}.$$

Finally, we multiply by f_p^{-1} and use the fact (3) that $f_p^{-1} \circledast f \equiv 1 \pmod{p}$ to retrieve the message

$$b \circledast f_p^{-1} \equiv m \circledast f \circledast f_p^{-1} \equiv m \pmod{p}.$$

In summary, we have shown in this section that the NTRU decoding procedure described in section 1.3 will recover the encoded message, subject to the probabilistic assumption that s lies in a gap. If s does not lie in the gap, then we will still be able to recover the message using $s \pm \delta$ in place of s for some (small) value of δ, since one of these values will lie in the gap as long as the gap exists. So the only situation in which the message is unrecoverable is when the A_k coefficients span an interval whose length is greater than $q-1$. For appropriate parameter choices, such decoding failure will be an extremely rare occurence. We will give a statistical analysis in the next section.

3. Statistical analysis

In this section we will analyze the probability that NTRU will fail to decode an encoded message. To illustrate our theoretical discussion, we will compute the probability of decoding failure for binary (N, p, q, d) NTRU with

$$N = 167, \quad p = 15, \quad q = 1024, \quad d = 71. \tag{9}$$

We begin with some notation. For any $\ell \geq 2$, we write B_ℓ for the space of polynomials of degree $N-1$ whose coefficients are between 0 and $\ell - 1$,

$$B_\ell = \left\{ \sum_{k=0}^{N-1} \epsilon_k x^k \;:\; 0 \leq \epsilon_k < \ell \right\}.$$

For any $F \in B_\ell$, we let

$$|F| = F(1) = (\text{sum of the coefficients of } F),$$

and we will let

$$B_\ell(d) = \{F \in B_\ell \;:\; |F| = d\}.$$

Our first step is to analyze the distribution of the coefficents of a product $f_1 \circledast f_2$ of binary polynomials. Let $f_1 \in B_2(d_1)$ and $f_2 \in B_2(d_2)$. If we write $f_1 = [a_1, \ldots, a_N]$ and $f_2 = [b_1, \ldots, b_N]$, then the k^{th} coefficient of $f_1 \circledast f_2$ is a sum of N terms of the form $a_i b_j$. We know that d_1 of the coefficients of f_1 are ones, so we can view this sum as picking d_1 coefficients of f_2 and adding them up. Notice, however, that we are choosing the d_1 coefficients of f_2 without replacement, so we get a hypergeometric distribution (see, e.g., [4, II §6]). Hence if we define a random variable

$$\begin{array}{rccc} X = X_{d_1,d_2,k} : & B_2(d_1) \times B_2(d_2) & \longrightarrow & \mathbb{Z}, \\ & (f_1, f_2) & \longmapsto & k^{\text{th}} \text{ coefficient of } f_1 \circledast f_2 \end{array}$$

then the distribution function of X will be (approximately)

$$\operatorname{Prob}(X = u) = \binom{N-d_2}{d_1-u}\binom{d_2}{u} \Big/ \binom{N}{d_1}.$$

(Yes, this is symmetric in d_1 and d_2.) We say approximately because the N coefficients of $f_1 \circledast f_2$ are not independent, they satisfy one relation, namely their sum is $d_1 d_2$. We expect that this small amount of correlation will only help us, and experiment seems to bear this out. In any case, the effect is negligible.

Next we consider the distribution of the coefficients of

$$\pi + f_1 \circledast f_2,$$

where $\pi = \pi_{f_1,f_2,p} \in B_p$ is chosen so that all of the coefficients of $\pi + f_1 \circledast f_2$: divisible by p. We define a new random variable

$$\begin{aligned} X' = X'_{d_1,d_2,p,k}: \quad B_2(d_1) \times B_2(d_2) &\longrightarrow \mathbb{Z}. \\ (f_1, f_2) &\longmapsto k^{\text{th}} \text{ coefficient of } \pi_{f_1,f_2,p} + f_1 \circledast f_2 \end{aligned}$$

Clearly we have

$$\operatorname{Prob}(X' = u) = \begin{cases} 0 & \text{if } p \nmid u, \\ \operatorname{Prob}(u - p < X \le u) & \text{if } p \mid u. \end{cases}$$

Thus our model for the coefficients of $\pi + f_1 \circledast f_2$ says that if $p|u$, then

$$\operatorname{Prob}(X' = u) = \binom{N}{d_1}^{-1} \cdot \sum_{k=0}^{p-1} \binom{N - d_2}{d_1 - u + k}\binom{d_2}{u - k}.$$

The following tables give the expected values for two parameter choices, incl ing (9), where the last column gives the likely number of coefficients having t value u.

u	$\mathrm{P}(X' = u)$	$N * \mathrm{P}(X' = u)$
0	0.000%	0.00
5	0.048%	5.10
10	0.716%	76.66
15	0.234%	25.09
20	0.001%	0.15
25	0.000%	0.00

$(N, p, d) = (107, 5, 31)$

u	$\mathrm{P}(X' = u)$	$N * \mathrm{P}(X' = u)$
15	0.000%	0.00
30	0.540%	90.19
45	0.460%	76.80
60	0.000%	0.00

$(N, p, d) = (167, 15, 71)$

Notice how the coefficients of $\pi + f \circledast g$ tend to be quite tightly clustered. On t other hand, there is enough variation that there is virtually no chance of finding t value of $\pi + f \circledast g$ by trial-and-error. For example, even if one knows that $N = 1$ and that $\pi + f \circledast g$ has 90 coefficients equal to 30 and 77 coefficients equal to there are still $\binom{167}{77} \approx 2^{162}$ possibilities.

Our goal is to analyze the likelihood that a random encoded message will decodable, preferably using the precomputed value s and t. This means that should fix our keys $f, g \in B_2(d)$, let π be the parity polynomial, and study t distribution of the coefficients of the multiple $\circledast$-product

$$\phi \circledast (\pi + f \circledast g) + f \circledast m$$

for varying $\phi \in B_2(d)$ and $m \in B_p$.

From now on we will assume that f, g, and d have been fixed. Consider the random variable

$$\begin{aligned} Y = Y_{f,g,d,k} : \quad B_2(d) \quad &\longrightarrow \quad \mathbb{Z}, \\ \phi \quad &\longmapsto \quad k^{\text{th}} \text{ coefficient of } \phi \circledast (\pi + f \circledast g) \end{aligned}$$

We can view a coefficient of the product $\phi \circledast (\pi + f \circledast g)$ as a sum of d randomly chosen coefficients from $\pi + f \circledast g$. As usual, the d coefficients are chosen without replacement.

Since f and g are fixed, $\pi + f \circledast g$ has a fixed distribution of coefficients. Let

$$N_k = \text{Number of coefficients of } \pi + f \circledast g \text{ equal to } k.$$

These satisfy

$$N_k = 0 \text{ if } p \nmid k \qquad \text{and} \qquad \sum_k N_k = N.$$

We will let rp be the largest subscript with $N_{rp} > 0$.

The random variable Y chooses d coefficients of $\pi + f \circledast g$ without replacement and adds them up. Suppose that Y chooses n_k of the k coefficients. We need $\sum n_k = d$, since Y is supposed to choose d coefficients, and for this choice, the value of Y is $\sum kn_k$. Further, there are $\prod \binom{N_k}{n_k}$ ways for Y to make its choices. So we have (the classical) formula for a hypergeometric distribution [4, II §2] (note Y only takes on values divisible by p)

$$\operatorname{Prob}(Y = u) = \binom{N}{d}^{-1} \sum_{\substack{(n_0, n_p, \ldots, n_{rp}) \\ 0 \le n_{kp} \le N_{kp} \\ n_0 + n_p + \cdots + n_{rp} = d \\ pn_p + 2pn_{2p} + \cdots + rpn_{rp} = u}} \binom{N_0}{n_0}\binom{N_p}{n_p}\binom{N_{2p}}{n_{2p}} \cdots \binom{N_{rp}}{n_{rp}}. \tag{10}$$

Although this formula looks somewhat forbidding, it can be calculated exactly for reasonably tight distributions for $N_0, \ldots, N_{rp}$.

For example, consider the parameter values (9). A typical coefficient distribution for $\pi + f \circledast g$ is $[N_{30}, N_{45}] = [90, 77]$, so the sum (10) reduces to a single term,

$$\operatorname{Prob}(Y = u) = \begin{cases} 0 & \text{if } 15 \nmid u, \\ \binom{90}{213 - u/15}\binom{77}{u/15 - 142} \Big/ \binom{167}{71} & \text{if } 15 \mid u. \end{cases}$$

We would like all of the coefficients of $\phi \circledast (\pi + f \circledast g)$ to lie within an interval whose width is considerably shorter than q; more precisely, we want to find out how large to take q so as to ensure that this happens. Since there are $N = 167$ coefficients, it is reasonable to expect that all of the coefficients will lie between α and β with probability $\operatorname{Prob}(\alpha \le Y \le \beta)^{167}$. The following list gives some typical values centered around the mean value of $E(Y) = d(p-1)/2 + d^3/N \approx 2640.18$:

$\operatorname{Prob}(2490 \le Y \le 2789)^{167} = 70.2705633719050\%$ (width 300)
$\operatorname{Prob}(2440 \le Y \le 2839)^{167} = 99.1161035564493\%$ (width 400)
$\operatorname{Prob}(2390 \le Y \le 2889)^{167} = 99.9901998757667\%$ (width 500)
$\operatorname{Prob}(2340 \le Y \le 2939)^{167} = 99.9999513596643\%$ (width 600)
$\operatorname{Prob}(2290 \le Y \le 2989)^{167} = 99.9999999963417\%$ (width 700)
$\operatorname{Prob}(2240 \le Y \le 3039)^{167} = 99.9999999999993\%$ (width 800)

This table shows that for $q = 512$ we are likely to see some decoding failure, but looks like $q = 1024$ will work well.

Next we need to consider the distribution of the coefficients of $f \circledast m$, so define a new random variable

$$Y' = Y'_{f,k} : \quad \begin{array}{ccc} B_p & \longrightarrow & \mathbf{Z}, \\ f & \longmapsto & k^{\text{th}} \text{ coefficient of } f \circledast m \end{array}$$

Although Y' can be viewed as choosing and summing d coefficients of m witho replacement, the fact that m ranges over all polynomials in B_p means that coefficients are completely independent. Hence the replacement issue is moot. Ea coefficient of m has a $1/p$ chance of assuming each of the values $0, 1, \ldots, p-1$, we can explicitly compute the generating function for Y' as follows:

$$\begin{aligned}\sum_{u \geq 0} \operatorname{Prob}(Y' = u) T^u &= \left(\frac{1}{p} + \frac{1}{p}T + \frac{1}{p}T^2 + \cdots + \frac{1}{p}T^{p-1}\right)^d \\ &= \frac{1}{p^d}\left(\frac{1-T^p}{1-T}\right)^d \\ &= \sum_{u \geq 0}\left(\frac{1}{p^d}\sum_{j=0}^{\min(d,u/p)} (-1)^j \binom{d+u-pj-1}{d-1}\binom{d}{j}\right)T^u.\end{aligned}$$

Using this formula, it's easy to find the probability that the coefficients of $f \circledast m$ within a given interval. For example, taking our suggested parameter values (we find (remember we raise to the 167th power to get the probability that all of t coefficients of $f \circledast m$ lie within the specified interval):

$\operatorname{Prob}(397 \leq Y' \leq 596)^{167} = 37.500983646819\%$	(width 200)
$\operatorname{Prob}(347 \leq Y' \leq 646)^{167} = 99.476631477151\%$	(width 300)
$\operatorname{Prob}(297 \leq Y' \leq 696)^{167} = 99.999660095207\%$	(width 400)
$\operatorname{Prob}(247 \leq Y' \leq 746)^{167} = 99.999999981242\%$	(width 500)
$\operatorname{Prob}(197 \leq Y' \leq 796)^{167} = 99.999999999999\%$	(width 600)

We are finally ready to consider the polynomial $\phi \circledast (\pi + f \circledast g) + f \circledast m$ who gap determines the decipherability of the NTRU encoded message m. We let Z the random variable describing the coefficients of $\phi \circledast (\pi + f \circledast g) + f \circledast m$,

$$Z = Z_{f,g,d,k} : \quad \begin{array}{ccc} B_2(d) \times B_p & \longrightarrow & \mathbf{Z}, \\ (\phi, m) & \longmapsto & k^{\text{th}} \text{ coefficient of } \phi \circledast (\pi + f \circledast g) + f \circledast m \end{array}$$

There should be little (if any) correlation between the coefficients of $f \circledast m$ and t coefficients of $\phi \circledast (\pi + f \circledast g)$, so we can write

$$Z = Y + Y'$$

and treat Y and Y' as independent variables. Then

$$\operatorname{Prob}(Z = u) = \sum_{u_1 + u_2 = u} \operatorname{Prob}(Y = u_1)\operatorname{Prob}(Y' = u_2).$$

Combining this with the formulas for $\text{Prob}(Y = u)$ and $\text{Prob}(Y' = u)$ given a
this gives a method for computing the distribution function of Z.

Now we want to determine if the precomputed gap value s correctly decod
message. This will be true if and only if all of the coefficients of $\phi \circledast (\pi + f \circledast g) +$
lie between $s + (Q-1)q$ and $s + Qq$, where Q is defined by (6). Assumin
the coefficients are independent (if anything, they will be slightly correlated,
will help), the probability that all N coefficients lie in the desired range is

$$\text{Prob}(s \text{ decodes } e) = \text{Prob}(s + (Q-1)q \le Z < s + Qq)^N.$$

Combining this with the formulas given above allows one to compute the proba
of decoding failure for any particular choice of f and g and choice of coeff
distribution for m.

For our suggested parameters (9), we have $s = 577$ and $Q = 3$, so

$$\text{Prob}(s \text{ decodes } e) = \text{Prob}(2625 \le Z < 3649)^{167}.$$

Rather than computing this quantity exactly, we will merely observe that

$$\text{Prob}(2625 \le Z < 3649) \ge$$
$$1 - \sum_{n=0}^{10} \text{Prob}(|Y - 2640| > 50n) \cdot \text{Prob}(|Y' - 497| > 1000 - 10$$

The sum is easily computed using the values in the earlier tables, and one
that

$$\text{Prob}(2625 \le Z < 3649)^{167} > 1 - 10^{-14}.$$

This shows that for the sample parameter values (9), the decoding process w
successful in virtually all cases.

Remark. The expected range of the coefficients of $\phi \circledast (\pi + f \circledast g) + f \circledast m$ de
only on the three parameters (N, d, p), and the message will be successfully de
provided that this range is smaller than the parameter q. This means that on
choose (N, d, p) as desired for purposes of security, and then choose q large e
to ensure that virtually all messages will be decodable. Of course, as q incr
there is some increase in computation (since we need to perform some div
by q with remainder), and the message expansion will increase; but the expans
proportional to $\log(q)$, so it is not greatly affected by small increases in q. How
we should mention that if one chooses q ridiculously large, there exists a possi
that a lattice reduction attack might be successful, see section 6.4.

Remark. If the precomputed value of s should fail to decode the message,
virtually certain that a nearby value will work. The most likely cause of dec
failure is that either the message check sum $|m|$ or the parity polynomial chec
$|\pi|$ is far from the expected values of $N(p-1)/2$, in which case the precompu
may fail to lie in the gap. In theory, for binary (N, p, q, d) NTRU, the best v
to use for s and t are

$$s \equiv \left\lfloor \frac{q}{2} + \frac{d(|\pi| + |m| + d^2)}{N} \right\rceil \pmod{q}, \qquad \text{with } 0 \le s < q.$$

$$t \equiv q \left\lfloor \frac{d(|\pi| + |m| + d^2)}{qN} \right\rceil \pmod{p}, \qquad \text{with } 0 \le t < p.$$

Thus Dan should probably check that his $|\pi|$ is close to $N(p-1)/2$, and if it isn't, then that (f, g) pair should be discarded. Similarly, it might be worthwhile for Cathy to transmit the value of $|m|$ together with the encoded message e. Then Dan can use $|m|$ as a check sum to validate the message, and also use it to adjust s if $|m|$ happens to significantly differ from the expected value.

Finally, we should mention the unhappy possibility that, in principle, there might be no value of s which will decode the message. This *gap failure* will occur if the coefficients of $\phi \circledast (\pi + f \circledast g) + f \circledast m$ span an interval whose width is larger than q. Notice that Cathy cannot check for this condition, since she doesn't know f or g. It is possible to analyze this situation theoretically, but we will be content here to note that for our suggested parameter values (9), the probability of this occurring is many orders of magnitude smaller than the probability that a message packet will become garbled in transmission; so from a practical standpoint, any standard error detecting/correcting method used to deal with transmission errors should be more than sufficient.

4. NTRU VARIANTS AND GENERALIZATIONS

4.1 Multiple transmissions of a single message. The NTRU cryptosystem as described in section 1 is highly susceptable to attack if a single message m is transmitted several times using the same public key h. As we will describe below, if Cathy transmists a message m several times using different fuzz values, then it is highly probable that a code breaker Betty will be able to recover the message m without knowing the private key f. More precisely, if Cathy uses (N, p, q, d) NTRU to transmit the message t times using different fuzz values $\phi_1, \ldots, \phi_t$, then Betty will be able to determine approximately $d(1-(d/N)^{t-1})$ of the d coefficients of ϕ_1 which equal one. For example, if $N = 167$, $d = 71$, and Cathy sends the message 5 times, Betty can probably recover the location of 68 of the one coefficients of ϕ_1. It would then be possible for her to try all $\binom{99}{3} = 156849$ possibilities for the other 3 one coefficients, thereby determining ϕ_1, and with it the message

$$m \equiv e_1 - \phi_1 \circledast h \pmod{q}.$$

Note, however, that although Betty will have decoded the message m, this in no way threatens the security of any other messages sent with the same key, since recovery of m does not help her to determine the decoding key f.

Before explaining how to securely retransmit messages, we mention that there are many situations, such as key exchange and identity verification, where a given message is never transmitted more than once. In such situations, the basic NTRU system described in section 1 will suffice. Further, if it is necessary to occasionally retransmit a message, say when noise in the line causes a detectable error, then one should use multi-NTRU only when retransmission is necessary. However, there are certainly situations, such as submission of credit card numbers over the Internet, where an identical message might be repeated at various times using a single public key and different fuzz values.

The following encryption method, which we will call the *multi-NTRU system*, allows safe multiple transmission of a single message using a single encoding key. In order to encode a message m using the public key h, Cathy chooses three random binary polynomials ϕ, ϕ', ζ, subject to the further requirement that ζ be $\circledast$-invertible modulo p. (Notice that this is the same as one of the requirements on the decoding key f.) Then she encodes the two messages ζ and $\zeta * m$ and transmits them. In other words, the message m is encoded by the two polynomials

$$e \equiv \phi \circledast h + \zeta \pmod q \qquad \text{and} \qquad e' \equiv \phi' \circledast h + \zeta \circledast m \pmod q.$$

When Dan recievies e and e', he uses his private key f to decode them as usual, recovering the mod p polynomials ζ and $\zeta \circledast m$. He next computes the inverse ζ^{-1} mod $(p, x^N - 1)$, and then the original message is recovered by multiplying

$$(\zeta \circledast m) \circledast \zeta^{-1} \equiv m \pmod p.$$

The multi-NTRU system has drawbacks for both encoder and decoder. In order to encode one message packet, Cathy must generate one random polynomial and perform two NTRU encodings, thereby doubling message expansion. To decode, Dan must do two NTRU decodings, plus invert a mod p polynomial. The advantage of multi-NTRU is that a single message may be transmitted any number of times with no loss of security.

We make two final comments. First, if Cathy owns the more powerful computational engine, she can make Dan's job easier by computing ζ^{-1} mod $(p, x^N - 1)$ and then sending the two messages ζ^{-1} and $\zeta \circledast m$. To recover m, Dan will then merely have to multiply the two decoded messages. In other words, the multi-NTRU system requires the computation of an inverse modulo p, but this inverse may be computed by either the encoder or the decoder.

Second, we think it is likely that knowledge of the binary polynomial ζ will not enable a codebreaker to decode the message. In other words, if Cathy sends the encoded message $e \equiv \phi \circledast h + \zeta \circledast m \mod q$ and also sends the unencoded value of ζ, the message will still be secure. Further, multiple transmissions using different pairs (ϕ_i, ζ_i) will remain secure even if the ζ_i's are made public.

As an application of this last remark, consider a situation such as credit card submissions over the Internet. Dan could precompute a list of random polynomials $\zeta_1, \zeta_2, \dots, \zeta_r$ and their mod p inverses. When Cathy wants to submit her credit card number, Dan sends her a randomly chosen ζ_i to use for multi-NTRU encoding. If r is taken to be of a reasonable size (e.g. 100 to 1000), and if a new list of ζ_i's is generated each day, it is extremely unlikely that Cathy would ever send the same message twice. In this way one gets the security of the multi-NTRU cryptosystem with little extra overhead.

For completeness, we briefly describe how Betty can recover a message m which has been encoded (but not multi-encoded) using the public key h and several different fuzz values $\phi_1, \dots, \phi_t$. Let $e_1, \dots, e_t$ be the encoded messages, so

$$e_i \equiv \phi_i \circledast h + m \pmod q.$$

Note that h, the public key, is assumed to be known, so Betty can compute its inverse modulo $(q, x^N - 1)$. The idea is to look at the differences

$$\phi_1 - \phi_i \equiv (e_1 - e_i) \circledast h^{-1} \pmod q, \qquad 2 \le i \le t.$$

For example, consider $\phi_2 - \phi_1$. The ϕ_i's are binary polynomials, so this difference will look like

$$\phi_1 - \phi_2 \equiv \sum_{k=0}^{N-1} \epsilon_k x^k \pmod{q, x^N - 1} \qquad \text{with } \epsilon_k \in \{-1, 0, 1\}.$$

Further, we have $\epsilon_k = 1$ if and only if the k^{th} coefficient of ϕ_1 equals 1 and the k^{th} coefficient of ϕ_2 equals 0. Since ϕ_1 has d one coefficients, and ϕ_2 has $N - d$ zero coefficients, Betty will probably learn the location of approximately $d(1 - d/N)$ of the one coefficients of ϕ_1.

Now suppose that Cathy transmits the message t times. Using the differences $e_1 - e_i$, each e_i will let Betty identify some of the remaining one coefficients of ϕ_1. Using t messages, she will on average recover $d(1 - (d/N)^{t-1})$ of the one coefficients. Thus even a small number of transmissions will allow her to recover ϕ_1 by a brute-force check of the remaining coefficients; and with ϕ_1 in hand, it is a simple matter to recover the message

$$m \equiv e_1 - \phi_1 \circledast h \pmod{x^N - 1}$$

without knowing the decoding key f.

On the other hand, suppose that Cathy has multi-encoded her messages, say $e_i \equiv \phi_i \circledast h + \zeta_i \circledast m \mod q$. Even if the ζ_i's are public knowledge, a codebreaker would only be able to determine the differences $\phi_1\zeta_i - \phi_i\zeta_1 \pmod{q}$. These values do not seem to be of much help in recovering the message.

4.2 Non-probablistic NTRU. The basic NTRU system described in section 1 is a probablistic public key cryptosystem in the sense of Goldwasser and Micali [5], since the message m is "concealed" by the application of the random fuzz ϕ in the encoded message $e = \phi \circledast h + m$. Clearly ϕ must be chosen in such a way that a potential eavesdropper cannot predict its value. On the other hand, it should be noted that Dan, the owner of the private key f, is able to recover both the message m and the value of the fuzz ϕ, since once he know m, then $\phi = (e - m) \circledast h^{-1} \mod q$. In principle, this means that Cathy could encode part of her message in ϕ; i.e., to encode her message, she could use N bits of her message to form ϕ, and the next $N \log_2(p)$ bits of her message to form m. This would improve the bandwidth, at the cost of some extra computation on Dan's part. Another way to make NTRU deterministic is to create ϕ by taking the message $m = [m_1, m_2, \ldots, m_N]$, reversing its coefficients, and reducing them modulo 2. In other words,

$$\phi = [m_N \bmod 2, m_{N-1} \bmod 2, \ldots, m_2 \bmod 2, m_1 \bmod 2].$$

Of course, one would need to be careful that the bits of Cathy's message are sufficiently random to use in ϕ. Further, the number of 1 bits in ϕ would now vary from message block to message block, so Cathy would need to transmit this number.

4.3 NTRU with 0% decoding failure. It is possible to eliminate gap failure entirely by choosing the parameter q sufficiently large. This also has the effect of making $s = t = 0$, so the "shifting step" in decoding is also eliminated. The simplest way to eliminate gap failure is to choose q so that the maximum range between the smallest coefficient and the largest coefficient of

$$\phi \circledast (\pi + f \circledast g) + f \circledast m$$

is smaller than q. The exact value of this maximum (as a function of (N,p,q,d) or (N,d,p,q,r)) looks like an extremely difficult combinatorial problem, but it's easy to give an upper bound, which suffices for our (theoretical) purposes. For example, a trivial analysis shows that the coefficient range is less than $d^2 + 2dp$ for binary NTRU and less than $r^5d^2 + r^2dp$ for symmetric NTRU. So if we choose q larger than this bound, gap failure disappears. A more careful analysis allows one to take considerably smaller values of q.

However, we note that from a practical viewpoint, there are two drawbacks to using such large q's. First, the public key and message expansion factors will become quite large. Second, if q is too large, then there is at least the possibility that some souped-up lattice reduction attack might be feasible (see section 6.4). For this reason, we would not normally recommend taking such large values of q.

4.4 A theoretical description of NTRU-type cryptosystems. If we ignore the possibility of decoding failure (or choose q sufficiently large so as to eliminate gap failure, see section 4.3), then NTRU is a "probabilistic public-key cryptosystem," a concept invented by Goldwasser and Micali ([5], see also [8, section 12.4]). If we slightly generalize the Goldwasser-Micali concept so as to allow for (occasional) decoding failure, we come up with the following definition.

Definition. A *probabilistic public-key cryptosystem* is a 6-tuple $(\mathcal{P},\mathcal{C},\mathcal{K},E,D,\mathcal{R})$, where $\mathcal{P}$ is the set of plaintexts, $\mathcal{C}$ is the set of ciphertexts, $\mathcal{K}$ is the keyspace, and E and D are encryption and decryption maps

$$E : \mathcal{K} \longrightarrow \mathrm{Map}(\mathcal{P}\times\mathcal{R},\mathcal{C}) \qquad \text{and} \qquad D : \mathcal{K} \longrightarrow \mathrm{Map}(\mathcal{C},\mathcal{P})$$

(i.e., for each $K \in \mathcal{K}$, we get an encryption rule $E_K : \mathcal{P}\times\mathcal{R} \to \mathcal{C}$ and a decryption rule $D_K : \mathcal{C} \to \mathcal{P}$), and $\mathcal{R}$ is a set of random fuzzes.

The *success rate* of a probabilisitic public-key cryptosystem is the number

$$\sigma = \sigma(\mathcal{P},\mathcal{C},\mathcal{K},e,d,\mathcal{R}) = \min_{\substack{K\in\mathcal{K}\\ b\in\mathcal{P}}} \frac{\#\{r \in \mathcal{R} : D_K(E_K(b,r)) = b\}}{\#\mathcal{R}}.$$

This is essentially the probability that the decryption rule D_K will decode a given encoded message. The system is called *complete* if $\sigma = 1$ (i.e., if $D_K(E_K(b,r)) = b$ for every $(b,r) \in \mathcal{P}\times\mathcal{R}$). We define the success rate and completeness of a particular key K in a similar manner.

There are (complete) probabilistic public-key cryptosystems due to Goldwasser-Micali [5] and to Blum-Goldwasser [1] which are based on the problem of finding square (respectively two-power) roots modulo pq for large primes p and q. See [8, section 12.4]. These systems are quite interesting from a theoretical viewpoint, but their speed seems (at best) comparable to RSA. NTRU, on the other hand, is a probabilisitic public-key cryptosystem which is considerably faster than RSA.

Abstractly, an NTRU-type cryptosystem can be defined as follows:

Definition. A set of *NTRU parameters* is a 6-tuple $(A, \mathfrak{p}, \mathfrak{q}, \mathcal{P}, \mathcal{Q}, \mathcal{R})$, where A is a ring, $\mathfrak{p}$ and $\mathfrak{q}$ are ideals in A, and $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{R}$ are subsets of A such that

$$\mathcal{P} \xrightarrow{\sim} A/\mathfrak{p} \qquad \text{and} \qquad \mathcal{Q} \xrightarrow{\sim} A/\mathfrak{q}.$$

(In other words, $\mathcal{P}$ is a set of coset representatives for $A/\mathfrak{p}$, and similarly for $\mathcal{Q}$.)

An *NTRU cryptosystem* for the parameters $(A, \mathfrak{p}, \mathfrak{q}, \mathcal{P}, \mathcal{Q}, \mathcal{R})$ is created as follows. Choose $f, g \in \mathcal{P}$ so that f has inverses modulo $\mathfrak{p}$ and modulo $\mathfrak{q}$, say $f_\mathfrak{p}^{-1} \in \mathcal{P}$ and $f_\mathfrak{q}^{-1} \in \mathcal{Q}$. That is,

$$f \circledast f_\mathfrak{p}^{-1} \equiv 1 \pmod{\mathfrak{p}} \qquad \text{and} \qquad f \circledast f_\mathfrak{q}^{-1} \equiv 1 \pmod{\mathfrak{q}},$$

where we write $\circledast$ for multiplication in A. Next choose $\pi \in \mathcal{P}$ so that

$$\pi + f \circledast g \equiv 0 \pmod{\mathfrak{p}},$$

and choose $h \in \mathcal{Q}$ so that

$$h \equiv \pi \circledast f_\mathfrak{q}^{-1} + g \pmod{\mathfrak{q}}.$$

(Note that once we pick f and g, there is exactly one choice for each of $f_\mathfrak{p}^{-1}$, $f_\mathfrak{q}^{-1}$, π and h.) The pair (f, h) is the desired private/public key pair.

The encoding rule E_h is given by

$$E_h : \mathcal{P} \times \mathcal{R} \longrightarrow \mathcal{Q}, \qquad E_h(b, r) \equiv r \circledast h + b \pmod{\mathfrak{q}}.$$

The decoding rule D_f is given by

$$D_f : \mathcal{Q} \longrightarrow \mathcal{P}, \qquad D_f(c) \equiv (c \circledast f \bmod \mathfrak{q}) \circledast f_\mathfrak{p}^{-1} \pmod{\mathfrak{p}},$$

where we write "$c \circledast f \bmod \mathfrak{q}$" to mean the representative in $\mathcal{Q}$ for the congruence class of $c \circledast f$ modulo $\mathfrak{q}$. More generally, the decoding rule may require a *shift function* $s : \mathcal{Q} \to A$, in which case the decoding rule is

$$D_f(c) \equiv s(c \circledast f \bmod \mathfrak{q}) \circledast f_\mathfrak{p}^{-1} \pmod{\mathfrak{p}}.$$

The key (f, h) will be complete (i.e., will decode every message) if the set

$$\{r \circledast (\pi + f \circledast g) + f \circledast b \, : \, b \in \mathcal{P}, \ r \in \mathcal{R}\} \quad \text{is contained in} \quad \mathcal{Q}.$$

If there is a shift function, (f, h) will be complete if

$$(r \circledast (\pi + f \circledast g) + f \circledast b \bmod \mathfrak{q}) = r \circledast (\pi + f \circledast g) + f \circledast b \qquad \text{for all } b \in \mathcal{P}, \ r \in \mathcal{R}.)$$

In general, the success rate of the key (f, h) is

$$\sigma(f, h) = \min_{b \in \mathcal{P}} \frac{\#\{r \in \mathcal{R} \, : \, r \circledast (\pi + f \circledast g) + f \circledast b \in \mathcal{Q}\}}{\#\mathcal{R}},$$

and similarly when there is a shift function.

Example. The "standard" (binary) NTRU system has:

$$A = \mathbb{Z}[X]/(X^N - 1), \qquad \mathfrak{p} = pA, \qquad \mathfrak{q} = qA,$$

$$\mathcal{P} = \left\{ \sum_{i=1}^{N} a_i X^{N-i} \,:\, 0 \le a_i < p \right\}$$

$$\mathcal{Q} = \left\{ \sum_{i=1}^{N} b_i X^{N-i} \,:\, 0 \le b_i < q \right\}$$

$$\mathcal{R} = \left\{ \sum_{i=1}^{N} c_i X^{N-i} \,:\, 0 \le c_i < 2 \text{ with exactly } d \text{ ones} \right\}$$

A similar system can be constructed by replacing $\mathbb{Z}$ with some other convenient ring. For example, take $A = \mathbf{F}_\ell[T][X]/(X^N - 1)$ and p and q polynomials in $\mathbf{F}_\ell[T]$. One could also consider variants of standard NTRU by using rings such as

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

This would slow computations somewhat, while providing greater mixing of the coefficients.

An NTRU-type cryptosystem is based on the algebraic operations in a ring A; that is, it uses both the addition and the multiplication in A. This stands in marked contrast to most other public key cryptosystems, such as the RSA and ElGamel (Discrete Log) systems which use a single group operation (generally modular multiplication or the group law on an elliptic curve) and knapsack (subset sum) systems which uses the monoid of positive integers under addition. We thus might call NTRU a *ring-based cryptosystem* to distinguish it from earlier *group-based cryptosystems* such as RSA and ElGamel.

Remark. The $\circledast$-multiplication used in NTRU can also be described using multiplication of circulant matrices, in which case NTRU bears a superficial resemblence to the McEliece public key cryptosystem [6]. The McEliece system using a matrix multiplication $HM + R$, where H is a rectangular matrix derived from an error correcting code (e.g., a Goppa code), the vector M is the message, and R is a random vector which is small enough to be dealt with by the error correction facility. A matrix formulation of NTRU looks like $HR + M$, where H is a square circulant matrix (i.e., a matrix whose rows are obtained by rotating the top row cyclically), R is a random vector, and M is the message. Although the encoding processes for McEliece's system and NTRU look similar, the decoding processes are entirely different. Further, the McEliece public key H is huge, since all of its entries must be specified; while an NTRU public key H is much smaller, since only its top row is needed.

5. Implementation considerations

In this section we analyze the choice of NTRU parameters, where we will concentrate on binary (N, p, q, d) NTRU and symmetric (N, p, q, d, r) NTRU. These parameters should be chosen with the following guidelines in mind:

(1) The number of available public keys f is approximately

$$\binom{N}{d} = \frac{N!}{d!(N-d)!} \qquad \text{(binary)}$$
$$\frac{N!}{(N-2dr-1)!\cdot d!^{2r-1}(d+1)!} \qquad \text{(symmetric)},$$

so N, d and r should be chosen to prevent a code breaker from trying all possible keys, keeping in mind the meet-in-the-middle attack. So to achieve a security level of 2^{80}, the number of keys should be on the order of 2^{160}.

(2) It is important that $\gcd(q,p) = 1$. Although in principle NTRU will work without this requirement, in practice having $\gcd(q,p) > 1$ will decrease security. At the extreme range, if $p|q$, then (exercise) the encoded message e satisfies $e \equiv m \mod p$, so it is completely insecure.

(3) We want most f's to have inverses f_q^{-1} and f_p^{-1}, since otherwise it will be hard to create keys. A first necessary requirement is that $\gcd(f(1),pq) = 1$. This is automatic for symmetric NTRU, and similarly for binary NTRU if we choose d to satisfy $\gcd(d,pq) = 1$. Assuming this, virtually all f's will have the required inverses if we take N to be a prime and require that for each prime P dividing p and q, the order of P in $(\mathbb{Z}/N\mathbb{Z})^*$ is large, say either $N-1$ or $(N-1)/2$. For example, this will certainly be true if $(N-1)/2$ is itself prime (i.e., N is a Sophie Germain prime). So especially good values for N include 83, 107, and 167.

(4) In order to ensure that $\pi + f \circledast g$ is not (approximately) a multiple of $(x^N-1)/(x-1)$, we should choose N, d, and p so that d^2/N is fairly close to a multiple of p. (For small p's, such as $p=2$ or 3, this requirement may be ignored.)

Let us briefly discuss item (3), the existence of the inverses f_q^{-1} and f_p^{-1}. We note that if these inverses exist, they can be rapidly computed using the Euclidean algorithm (see [2, §1.3.2] or section 7). Whether they exist depends, as one might expect, on f, q, and p. In general, a monic polynomial f will have a $\circledast$-inverse modulo M if (and only if) for every prime P dividing M, we have

$$\gcd(f, x^N - 1) = 1 \quad \text{in } (\mathbb{Z}/P\mathbb{Z})[x]. \tag{11}$$

In particular, since $x-1$ divides x^N-1, a necessary condition for f to have a $\circledast$-inverse modulo M is that $\gcd\bigl(f(1), M\bigr) = 1$. Assuming this, the probability that a randomly chosen f will satisfy (11) depends on the factorization of $\Lambda_N = x^{N-1} + x^{N-2} + \cdots + x + 1$ modulo P; the more (small) factors it has, the more likely it is that f will have a factor in common. Now if N is prime and if P^e is the smallest power of P which is congruent to 1 modulo N, then it's not hard to see that Λ_N has $(N-1)/e$ irreducible factors each of degree e modulo P. A particularly good situation occurs if $(N-1)/2$ also happens to be prime, since then the only choices for e are 1, 2, $(N-1)/2$ and $N-1$. So if $P^2 \not\equiv 1 \pmod{N}$, as will be the case in practice, then virtually every f will have an inverse modulo P.

We must also ask if e contains any hints to help in determining ϕ and/or m. One way to think of the product $\phi \circledast h$ is to consider the coefficients of h as lying in a cyclical list:

$$\dots, h_{N-1}, h_N, h_1, h_2, h_3, \dots, h_N, h_1, h_2, \dots.$$

As noted above, these coefficients will be randomly distributed between 1 and $q-1$. We now randomly choose d (different) starting points $i, j, k, \dots, \ell$ and take N consecutive entries starting at each one. After that, we list the coefficients of the message, add the columns, and reduce modulo q to get the encoded message:

$$\begin{array}{cccccc}
 & h_i & h_{i+1} & h_{i+2} & \cdots & h_{i+N-1} \\
 & h_j & h_{j+1} & h_{j+2} & \cdots & h_{j+N-1} \\
 & h_k & h_{k+1} & h_{k+2} & \cdots & h_{k+N-1} \\
 & \vdots & & & \ddots & \vdots \\
 & h_\ell & h_{\ell+1} & h_{\ell+2} & \cdots & h_{\ell+N-1} \\
+ & m_N & m_{N-1} & m_{N-2} & \cdots & m_1 \\
\hline
 & e_1 & e_2 & e_3 & \cdots & e_N
\end{array} \pmod q$$

The randomness of the coefficients of h will drown out the effect of m in these sums modulo q, so the encoded message e will not obviously reflect either the message m or the starting points $i, j, k, \dots, \ell$.

6.2 Meet-in-the-middle attacks. As pointed out by Andrew Odlyzko, there is a meet-in-the-middle attack which can be used against the fuzz ϕ. We observe that a similar attack applies also to the private key f. Briefly, one splits f in half, say $f = f_1 + f_2$, and then one matches $f_1 \circledast e$ against $-f_2 \circledast e$, looking for (f_1, f_2) so that the corresponding coefficients have approximately the same value. Hence in order to obtain a security level of (say) 2^{80}, one should choose f, g, and ϕ from a set containing around 2^{160} elements.

6.3 Multiple transmission attacks. As already noted in section 4.1, multiple transmissions of a single message m using different fuzzes will likely compromise the security of m, although it will not affect the security of the private key f.

6.4 Lattice reduction attacks. Finally, we should ask if there might be an attack on NTRU using lattice reduction techniques. The natural way to formulate such an attack is to consider the lattice L generated by the columns of the following $2N \times 2N$ matrix:

$$M = \left(\begin{array}{cccc|cccc}
1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\
\hline
h_1 & h_2 & \cdots & h_N & q & 0 & \cdots & 0 \\
h_2 & h_3 & \cdots & h_1 & 0 & q & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
h_N & h_1 & \cdots & h_{N-1} & 0 & 0 & \cdots & q
\end{array}\right)$$

We must also ask if e contains any hints to help in determining ϕ and/or m. One way to think of the product $\phi \circledast h$ is to consider the coefficients of h as lying in a cyclical list:

$$\dots, h_{N-1}, h_N, h_1, h_2, h_3, \dots, h_N, h_1, h_2, \dots .$$

As noted above, these coefficients will be randomly distributed between 1 and $q-1$. We now randomly choose d (different) starting points $i, j, k, \dots, \ell$ and take N consecutive entries starting at each one. After that, we list the coefficients of the message, add the columns, and reduce modulo q to get the encoded message:

$$\begin{array}{cccccc}
 & h_i & h_{i+1} & h_{i+2} & \cdots & h_{i+N-1} \\
 & h_j & h_{j+1} & h_{j+2} & \cdots & h_{j+N-1} \\
 & h_k & h_{k+1} & h_{k+2} & \cdots & h_{k+N-1} \\
 & \vdots & & & \ddots & \vdots \\
 & h_\ell & h_{\ell+1} & h_{\ell+2} & \cdots & h_{\ell+N-1} \\
+ & m_N & m_{N-1} & m_{N-2} & \cdots & m_1 \\
\hline
 & e_1 & e_2 & e_3 & \cdots & e_N
\end{array} \pmod q$$

The randomness of the coefficients of h will drown out the effect of m in these sums modulo q, so the encoded message e will not obviously reflect either the message m or the starting points $i, j, k, \dots, \ell$.

6.2 Meet-in-the-middle attacks. As pointed out by Andrew Odlyzko, there is a meet-in-the-middle attack which can be used against the fuzz ϕ. We observe that a similar attack applies also to the private key f. Briefly, one splits f in half, say $f = f_1 + f_2$, and then one matches $f_1 \circledast e$ against $-f_2 \circledast e$, looking for (f_1, f_2) so that the corresponding coefficients have approximately the same value. Hence in order to obtain a security level of (say) 2^{80}, one should choose f, g, and ϕ from a set containing around 2^{160} elements.

6.3 Multiple transmission attacks. As already noted in section 4.1, multiple transmissions of a single message m using different fuzzes will likely compromise the security of m, although it will not affect the security of the private key f.

6.4 Lattice reduction attacks. Finally, we should ask if there might be an attack on NTRU using lattice reduction techniques. The natural way to formulate such an attack is to consider the lattice L generated by the columns of the following $2N \times 2N$ matrix:

$$M = \left(\begin{array}{cccc|cccc}
1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\
\hline
h_1 & h_2 & \cdots & h_N & q & 0 & \cdots & 0 \\
h_2 & h_3 & \cdots & h_1 & 0 & q & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
h_N & h_1 & \cdots & h_{N-1} & 0 & 0 & \cdots & q
\end{array}\right)$$

Notice that $\det(M) = q^N$, so L has (co)volume q^N and rank $2N$. This means that L will contain many vectors of length $O(q^{1/2})$.

We know that the product $f \circledast h$, when reduced modulo q, has comparatively small coefficients. So suppose that we write

$$f_N h_k + f_{N-1} h_{k+1} + \cdots + f_2 h_{N+k-2} + f_1 h_{N+k-1} = qI_k + J_k$$

for each $1 \le k \le N$. The construction of h says that we can do this so that the J_k's are the coefficients of $\pi + f \circledast g$. So if we multiply the matrix M by the (column) vector

$$[f_N, f_{N-1}, \ldots, f_2, f_1, -I_1, -I_2, \ldots, -I_N],$$

we find that the lattice L contains the "small" vector

$$\mathbf{v} = [f_N, f_{N-1}, \ldots, f_2, f_1, J_1, J_1, \ldots, J_N].$$

If this vector is really a small vector of L, then we might be able to use lattice reduction (e.g., the L^3-algorithm) to find $\mathbf{v}$.

We can estimate the length of $\mathbf{v}$ as follows. For binary (N,p,q,d) NTRU we find:

(1) Average value of the f_i's is d/N.
(2) Average value of the J_k's is $(p-1)/2 + d^2/N$.

Thus

$$|\mathbf{v}| \approx \sqrt{N(d/N)^2 + N((p-1)/2 + d^2/N)^2}.$$

For any reasonable implementation, this will be considerably larger than $q^{1/2}$, so in fact one cannot find $\mathbf{v}$ (and with it, f) using lattice reduction techniques. To illustrate, here are the values for the three implementations described in Table 1.

(N,p,q,d)	$\sqrt{q}$	$\sqrt{N(d/N)^2 + N((p-1)/2 + d^2/N)^2}$
$(107,5,256,31)$	16.00	113.63
$(167,7,512,71)$	22.63	428.89
$(167,15,1024,71)$	32.00	480.58

Of course, $\mathbf{v}$ has the additional property that the J_k's are divisible by p, but this congruence property does not seem to help in performing a lattice reduction search.

Finally, rather than looking for small vectors in L, we might try to look for vectors in L which are close to the vector

$$\mathbf{w} = \Bigl[\underbrace{\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2}}_{N \text{ copies}}, \underbrace{\frac{p-1}{2} + \frac{d^2}{N}, \ldots, \frac{p-1}{2} + \frac{d^2}{N}}_{N \text{ copies}}\Bigr].$$

Notice $\mathbf{w}$ is the average or expected value of the true vector $\mathbf{v}$ that we are looking for. There are two important observations to be made concerning this line of attack.

First, we are now trying to find a vector in the lattice L which is close to the non-lattice vector $\mathbf{w}$. At the moment, there does not exist a good algorithm (e.g., as powerful as L^3) for solving this problem. The best current method is to use L^3 to find a small basis for L, write $\mathbf{w}$ as an $\mathbf{R}$-linear combination of the basis, round the real coefficients off to integers, and then try modifying the coefficients up and down. This is unlikely to work unless $\mathbf{w}$ is extremely close to a point of L.

Second, suppose that we grant some future breakthrough enables someone to solve this lattice proximity problem efficiently. Then NTRU will still be secure provided that the distance $|\mathbf{v}-\mathbf{w}|$ is (at least) as large as $q^{1/2}$, since then there will be exponentially (in N) many points of L which are just as close to $\mathbf{w}$ as $\mathbf{v}$ is, and there will be no way to pick out $\mathbf{v}$ from this cloud of points. So how close is $\mathbf{v}$ to $\mathbf{w}$? This depends on the distribution of the coefficients of $\pi + f \circledast g$. For example, if we take $(N,p,q,d) = (167,15,1024,71)$, then a typical distribution for $\pi + f \circledast g$ is $[N_{30}, N_{45}] = [90,77]$ (see section 3), in which case we have

$$|\mathbf{v}-\mathbf{w}| \approx 96.91 \qquad \text{and} \qquad q^{1/2} = 32.$$

This will certainly be safe. Similarly, if we use $(N,p,q,d) = (107,5,256,31)$, then $\pi + f \circledast g$ will have a coefficient distribution like $[N_5, N_{10}, N_{15}] = [5,77,25]$, and then

$$|\mathbf{v}-\mathbf{w}| \approx 25.63 \qquad \text{and} \qquad q^{1/2} = 16,$$

which again should be safe.

In conclusion, for appropriate choice of parameters, NTRU appears to be secure against lattice reduction methods, including any future progress in solving the lattice proximity problem.

6.5 Existence of multiple keys. One might ask if f is the only private key which can be used to decoded messages which have been encoded using the public key h. Any decoding key f' for h will need to have both of the following properties:

(1) When the coefficients of $f' \circledast h$ are first reduced modulo q and then shifted, they all become divisible by p.

(2) The coefficients of $f' \circledast m$ must lie in a short interval, for any message m.

While either of conditions (1) or (2) may be easily satisfied, it is hard to satisfy both simultaneously. To see this, we note that it appears that for f' to satisfy condition (2), its coefficients must all be "small", i.e., considerably smaller than $q/2$. But one would expect the coefficients of an f' satisfying (1) to be uniformly distributed between 0 and q. Thus the probability that an f' satisfying (1) will also satisfy (2) is (much) less than $(1/2)^N$. On the other hand, the probability that an f which satisfies (2) also satisfies (1) is approximately $(1/p)^N$.

7. Pseudo-code implementation

In this section we will briefly give pseudo-code implementing the main routines for binary (N,p,q,d) NTRU. For the purposes of computer implementation, it is most convenient to represent polynomials as vectors

$$\mathbf{a} = [a_1, a_2, \dots, a_N] \quad \text{corresponding to} \quad a(x) = \sum_{i=1}^{N} a_i x^{N-i}.$$

We will give pseudo-code for the following routines:

CREATEKEY(N,p,q,d)	Create a private/public key pair (f,h)
ENCODE(m,h,N,q,d)	Use h to encode a message m
DECODE(e,f,N,p,q,d)	Use f to decode a message e
STARMULTIPLY(a,b,N,M)	Multiply $a \circledast b$ (modulo M if $M > 0$)
RANDOMBINARYPOLY(N,d)	Create a random binary polynomial with d ones
INVERSEPOLY(a,N,M)	Compute the inverse of a modulo $(M, x^N - 1)$

We will leave the reader to supply their own routines for generating random numbers and for performing division with remainder in the polynomial ring $(\mathbb{Z}/P\mathbb{Z})[x]$ when P is prime. We also mention that the RANDOMBINARYPOLY routine we give is not the most efficient, but it is easy to implement and works well in practice. Of course, a slightly more complicated routine could be written to choose d coordinates without replacement.

```
CREATEKEY($N,p,q,d$)
(Create a public/private key pair $(f,h)$. Also return $f_p^{-1} = f^{-1}$ (mod $p$) for use
by the decoding routine.)
    $\mathbf{g}$ = RANDOMBINARYPOLY($N,d$)
    $\mathbf{f}$ = RANDOMBINARYPOLY($N,d$)
    $\mathbf{f}_q^{-1}$ = INVERSEPOLY($\mathbf{f},N,q$)
    $\mathbf{f}_p^{-1}$ = INVERSEPOLY($\mathbf{f},N,p$)
    IF $\mathbf{f}_q^{-1}$ OR $\mathbf{f}_p^{-1}$ DOES NOT EXIST, THEN CHOOSE A NEW $\mathbf{f}$
    $\mathbf{F}$ = STARMULTIPLY($\mathbf{f},\mathbf{g},N,p$)
    $\pi = -\mathbf{F}$ (mod $p$)
    $\mathbf{h}$ = STARMULTIPLY($\pi,\mathbf{f}_q^{-1},N,q$) + $\mathbf{g}$ (mod $q$)
    RETURN ($\mathbf{f},\mathbf{h},\mathbf{f}_p^{-1}$)
```

```
ENCODE($\mathbf{m},\mathbf{h},N,q,d$)
(Encode the message $m$ using the public key $h$.)
The following should be precomputed and stored:
    $\mathbf{h}' = [h_1,h_2,\dots,h_N,h_1,h_2,\dots,h_{N-1}]$
The ENCODE routine begins here:
    Choose $d$ distinct[†] random numbers $[k_1,k_2,\dots,k_d]$ between 1 and $N$
    $\mathbf{e} = \mathbf{m} + \sum_{i=1}^{d} [h'_{N+1-k_i}, h'_{N+2-k_i}, \dots, h'_{2N-k_i}]$ (mod $q$)
    RETURN $\mathbf{e}$
```

[†]For greater speed, one can drop the requirement that the k_i's be distinct. This will lead to a greater probability of decoding failure, but for reasonable parameters (N,p,q,d), decoding failure will still be extremely rare.

```
DECODE(e, f, $N, p, q, d$)
(Decode the message $e$ using the private key $f$.)
The following should be precomputed and stored:
    $\mathbf{u} = [u_1, \ldots, u_d]$ list of indices for which $f_{u_i} = 1$
    $s = \left\lceil \frac{q}{2} + d(p-1) + \frac{d^3}{N} \right\rceil \pmod{q}$
    $t = q \left\lceil \frac{d(p-1) + d^3/N}{q} \right\rceil \pmod{p}$
    $[v_1, v_2, \ldots, v_N] = \mathbf{f}_p^{-1} =$ INVERSEPOLY(f, $N, p$)
    $\mathbf{v}' = [v_1, v_2, \ldots, v_N, v_1, v_2, \ldots, v_{N-1}]$
The DECODE routine begins here:
    $\mathbf{e}' = [e_1, e_2, \ldots, e_N, e_1, e_2, \ldots, e_{N-1}]$
    $\mathbf{a} = \sum_{i=1}^{d} [e'_{N+1-u_i}, e'_{N+2-u_i}, \ldots, e'_{2N-u_i}] \pmod{q}$
    $\mathbf{b} = [b_1, b_2, \ldots, b_N]$ with $b_k = \begin{cases} a_k + t & \text{if } a_k < s, \\ a_k + t - q & \text{if } a_k \geq s. \end{cases}$
    $\mathbf{n} = \sum_{k=1}^{N} b_k * [v'_{N+1-k}, v'_{N+2-k}, \ldots, v'_{2N-k}] \pmod{p}$
    RETURN n
```

```
STARMULTIPLY(a, b, $N, M$)
(Compute $a(x) \circledast b(x)$ modulo $x^N - 1$, and modulo $M$ if $M > 0$. This implementation
requires $N^2$ multiplications. If either a or b is a binary polynomial, there are faster
implementations.)
    DO $k = 1$ TO $N$
        $c_k = 0$
        $j = k - 1$
        DO $i = 1$ TO $N$
            IF $j = 0$ THEN $j = N$
            $c_k = c_k + a_i * b_j$
            $j = j - 1$
        END $i$ LOOP
        IF $M > 0$ THEN $c_k = c_k \pmod{M}$
    END $k$ LOOP
    RETURN $\mathbf{c} = [c_1, c_2, \ldots, c_N]$
```

```
RANDOMBINARYPOLY($N, d$)
(Create a random polynomial (vector) with $d$ ones and $N - d$ zeros)
    $\mathbf{a} = 0$
    $k = d$
    WHILE $k > 0$
        $j =$ Random number between 1 and $N$ (inclusive)
        IF $a_j = 0$ THEN $a_j = 1$; $k = k - 1$
    END WHILE
    RETURN $\mathbf{a} = [a_1, a_2, \ldots, a_N]$
```

```
INVERSEPOLY(a, $N$, $P$, $r$)
(Compute the inverse of $a(x)$ modulo the ideal $(P^r, x^N - 1)$, where we assume
that $P$ is prime. The first part of this routine is adapted from [2, §1.3.2]. More
generally, to compute the inverse of $a(x)$ modulo $(M, x^N - 1)$, one first factors $M$
as $M = \prod P_i^{r_i}$, next computes the inverse $a_i(x)^{-1}$ modulo $(P_i^{r_i}, x^N - 1)$, and finally
combines the $a_i(x)^{-1}$'s using the Chinese remainder theorem to obtain $a(x)^{-1}$.)
(Compute the inverse of a modulo $P$)
    $\mathbf{b} = x^N - 1$;  $\mathbf{d} = \mathbf{a}$;  $\mathbf{u} = 1$;  $\mathbf{v}_1 = 0$;  $\mathbf{v}_3 = \mathbf{b}$;
    WHILE $\mathbf{v}_3 \neq 0$
        WRITE $\mathbf{d} = \mathbf{v}_3 * \mathbf{q} + \mathbf{t}_3 \pmod{P}$ WITH $\deg(\mathbf{t}_3) < \deg(\mathbf{v}_3)$
            (This is long division of polynomials in $(\mathbb{Z}/P\mathbb{Z})[x]$.)
        $\mathbf{t}_1 = \mathbf{u} - \mathbf{q} * \mathbf{v}_1$;  $\mathbf{u} = \mathbf{v}_1$;  $\mathbf{d} = \mathbf{v}_3$;  $\mathbf{v}_1 = \mathbf{t}_1$;  $\mathbf{v}_3 = \mathbf{t}_3$
    END WHILE
    IF $\deg(\mathbf{d}) > 0$ THEN RETURN 'Inversion Fails'
    $\mathbf{c} = d_0^{-1}\mathbf{u} \pmod{P}$
(If $r > 1$, refine to get an inverse modulo $P^r$)
    $Q = P$
    WHILE $Q < P^r$
        $\mathbf{c}' = (\text{STARMULTIPLY}(\mathbf{a}, \mathbf{c}, N, Q^2) - 1)/Q$
        $\mathbf{c} = \mathbf{c} - Q * \text{STARMULTIPLY}(\mathbf{c}, \mathbf{c}', N, Q)$
        $Q = Q^2$
    END WHILE
    RETURN $\mathbf{c} \pmod{P^r}$
```

References

1. M. Blum, S. Goldwasser, *An efficient probabilistic public-key encryption scheme which hides all partial information*, Advances in Cryptology: Proceedings of CRYPTO 84, Lecture Notes in Computer Science, vol. 196, Springer-Verlag, 1985, pp. 289–299.
2. H. Cohen, *A course in computational algebraic number theory*, Graduate Texts in Math., vol. 138, Springer Verlag, Berlin, 1993.
3. W. Diffie, M.E. Hellman, *New directions in cryptography*, IEEE Trans. on Information Theory 22 (1976), 644–654.
4. W. Feller, *An Introduction to Probability Theory and Its Applications*, Third Edition, vol. I, John Wiley & Sons, Inc., New York, 1968.
5. S. Goldwasser and A. Micali, *Probabilistic encryption*, J. Computer and Systems Science 28 (1984), 270–299.
6. R.J. McEliece, *A public-key cryptosystem based on algebraic coding theory*, JPL Pasadena, DSN Progress Reports 42–44 (1978), 114–116.
7. R.L. Rivest, A. Shamir, L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, Communications of the ACM 21 (1978), 120–126.
8. D. Stinson, *Cryptography: Theory and Practice*, CRC Press, Boca Raton, 1995.

Mathematics Department Box 1917 Brown University Providence, RI 02912 USA
E-mail address: jhoff@gauss.math.brown.edu

Mathematics Department Box 1917 Brown University Providence, RI 02912 USA
E-mail address: jpipher@gauss.math.brown.edu

Mathematics Department Box 1917 Brown University Providence, RI 02912 USA
E-mail address: jhs@gauss.math.brown.edu