NTRUSIGN: DIGITAL SIGNATURES USING THE NTRU LATTICE
PRELIMINARY DRAFT 2 — APRIL 2, 2002

JEFFREY HOFFSTEIN, NICK HOWGRAVE-GRAHAM, JILL PIPHER,
JOSEPH H. SILVERMAN, WILLIAM WHYTE

INTRODUCTION

Secure public key authentication and digital signatures are increasingly important for electronic communications, commerce, and security. They are required not only on high powered desktop computers, but also on devices with severely constrained memory and processing capabilities, such as smart cards, cell phones, and RFID tokens. The importance of public key authentication and digital signatures is amply demonstrated by the large literature devoted to both theoretical and practical aspects of the problem, see for example [1, 6, 11, 12, 19, 22, 26, 28, 29, 30].

At CRYPTO '96 a highly efficient new public key cryptosystem called NTRU was introduced. (See [8] for details.) Underlying NTRU is the hard mathematical problem of finding short and/or close vectors in a certain class of lattices, called convolution modular lattices or NTRU lattices. In this paper we present a complementary fast authentication and digital signature scheme, which we call NTRUSIGN, based on the same underlying hard problem in the same lattices used by NTRU. Henceforth the original NTRU public key encryption/decryption algorithm will be referred to as NTRUENCRYPT.

The core idea of NTRUSIGN is as follows. The Signer's private key is a short generating basis for an NTRU lattice and his public key is a much longer generating basis for the same lattice. (Notice that this is not quite the same as for NTRUENCRYPT where we just had one short generating vector.) The signature on a digital document is a vector in the lattice with two important properties:

- The signature is attached to the digital document being signed.
- The signature demonstrates an ability to solve a general closest vector problem in the lattice.

The way in which NTRUSIGN achieves these two properties may be briefly summarized as follows:

Key Generation: The private key includes a short $2N$-dimensional vector denoted (f, g). The public key is the large N-dimensional vector h that specifies the NTRU lattice L_h^{NT}, that is, h is generated from f and g by the usual NTRU convolution congruence $h \equiv f^{-1} * g \pmod{q}$. The private key also includes a complementary short vector (F, G) that is chosen so that (f, g) and (F, G) generate the full NTRU lattice L_h^{NT}.

1991 *Mathematics Subject Classification.* Primary: 94A60; Secondary: 11T71.

Key words and phrases. digital signature, public key authentication, NTRU, NTRUSIGN, NTRUENCRYPT, lattice-based cryptography, closest vector problem.

Signing: The digital document to be signed is hashed to create a random vector (m_1, m_2) modulo q. The signer uses the (secret) short generating vectors to find a lattice vector (s, t) that is close to (m_1, m_2).

Verification: The verifier uses the public key h to verify that (s, t) is indeed in the lattice L_h^{NT} and he verifies that (s, t) is appropriately close to (m_1, m_2).

Remark 1. An earlier digital signature scheme called NSS, also based on NTRU lattices, was presented at Eurocrypt 2001 [9]. A number of cryptographers found weaknesses in NSS due to the incomplete linkage between an NSS signature and the underlying hard lattice problem. See Section 7 for details. These difficulties have been solved in NTRUSIGN, since there is a direct and straightforward linkage between NTRUSIGN signatures and the (approximate) closest vector problem in the underlying NTRU lattice.

Remark 2. The principle upon which NTRUSIGN is based is very simple. The signer has private knowledge of short basis vectors in the NTRU lattice. Given an arbitrary point in space arising from a message digest, the signer uses this knowledge to find a point in the NTRU lattice close to the message point. He then exhibits this approximate solution to the closest vector problem (CVP) as his signature. This basic idea was already proposed by Goldreich, Goldwasser and Halevi in [5].

The fundamental advance in this paper is the use of NTRU lattices for CVP-based signatures. The cyclical nature of the NTRU lattices allows the public key to be specified by just one or two vectors, and it is this property that allows secure instances of NTRUENCRYPT and NTRUSIGN with practical key sizes. Thus for lattices of dimension n, the GGH proposal [5] requires keys of size $O(n^2)$ bits, while NTRUENCRYPT and NTRUSIGN use keys of size $O(n \log n)$ bits. At a practical level, this means that a secure version of GGH requires keys with between 10^5 and 10^6 bits (see [21]), while NTRUENCRYPT and NTRUSIGN achieve RSA 1024 bit security with keys of under 2000 bits. Thus NTRUENCRYPT and NTRUSIGN use RSA-size keys while achieving orders of magnitude speed and footprint advantages over RSA and ECC.

It should be noted that the use of NTRU lattices for CVP-based signatures is not completely straightforward. For the GGH scheme, the signer is free to choose any basis of short vectors as his private key. For an NTRU lattice, the first short vector (f, g) and the public parameters N and q completely determine the lattice L_h^{NT}, so the signer only has a short basis for half of the lattice. Thus he needs to use the known short vector (f, g) to find a complementary short vector (F, G) that, together with (f, g), generates L_h^{NT}. The efficient construction of an appropriate (F, G) is a nontrivial task; we describe theoretical and practical algorithms in Section 5.

We will show that forgery of a signature implies the ability to solve an (approximate) closest vector problem in high dimension for the class of NTRU lattices. We will also analyze the information contained in a long transcript of valid signatures and show that the generic method used in signature creation dramatically reduces the feasability of an attack from this direction. In particular, we will show that the most efficient known method of transcript analysis is ineffective on transcripts of a 10^8 valid signatures created with a single private key.

We start in Section 1 with a very brief description of NTRUSIGN. The remainder of the article (aside from the last section) then provides explanation, amplification, and justification for the signature scheme described in this initial section.

The body of the article thus begins in Section 2 with an overview of convolution modular lattices and the hard lattice problems that are used by NTRUENCRYPT and NTRUSIGN. In Section 3 we sketch the new aspects of the key creation process and give the signing and verification protocols for NTRUSIGN. We also provide specific parameter choices for a level of security equivalent to RSA 1024, and give preliminary timing results associated with these parameters. In Section 4 we analyze the security of NTRUSIGN. We show that creating a valid NTRUSIGN signature directly from the public key is closely tied to the classical closest vector problem, analyze the requirements on the hash function, and consider what information is available to an attacker who obtains a transcript of signatures. In Section 5 we complete the discussion of key creation and describe a method for finding a complete short basis for an NTRU lattice, given knowledge of the private vector (f, g). In Section 6 we describe some potential ways in which NTRUSIGN might be made even more efficient. Finally, in Section 7, we briefly review the history of an earlier NTRU lattice based signature scheme called NSS and contrast its ad hoc document encoding method via auxiliary congruences (which led to certain weaknesses) with the direct linkage of NTRUSIGN signatures to the underlying CVP.

This is a second and still preliminary draft of the paper describing NTRUSIGN. The first draft was distributed at the rump session of AsiaCrypt '01. This draft incorporates some helpful comments that were made by Craig Gentry, Jyrki Lahtonen, Ari Renvall and Mike Szydlo.

1. A Brief Overview of NTRUSIGN

We very briefly describe NTRUSIGN. Further details are provided in the remaining sections of this article. All polynomial products are modulo $X^N - 1$.

Public Parameters: Select a (prime) dimension N, a modulus q, key size parameters d_f and d_g, and a verfication bound parameter `NormBound`.

Key Generation: Choose binary polynomials f and g with d_f ones and d_g ones, respectively. Compute the public key $h \equiv f^{-1} * g \pmod q$. Compute small polynomials (F, G) satisfying $f * G - g * F = q$.

Signing: Hash the digital document D to create a random vector (m_1, m_2) mod q. Write

$$G * m_1 - F * m_2 = A + q * B,$$
$$-g * m_1 + f * m_2 = a + q * b,$$

where A and a have coefficients between $-q/2$ and $q/2$. The signature on D is the polynomial s given by

$$s \equiv f * B + F * b \pmod q.$$

Verification: Hash the digital document D to recreate (m_1, m_2). Compute $t \equiv s * h \pmod q$. Verify that

$$\|s - m_1\|^2 + \|t - m_2\|^2 \leq \texttt{NormBound}^2.$$

Suggested Parameters: The following parameters appear to offer security at least as great as that provided by RSA 1024, that is, an estimated breaking time greater than 10^{12} MIPS-years:

$$N = 251, \quad q = 128, \quad d_f = 73, \quad d_g = 71, \quad \texttt{NormBound} = 300.$$

2. NTRU Lattices and Associated Vector Problems

The NTRU Public Key Cryptosystem (NTRUEncrypt)and the related NTRU Signature Scheme (NTRUSign) use two public parameters N and q. Typical choices are $(N, q) = (251, 128)$ and $(N, q) = (503, 256)$. Basic operations take place in the ring of convolution polynomials

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

A polynomial $a(X) = a_0 + a_1X + \cdots + a_{N-1}X^{N-1} \in R$ is identified with its vector of coordinates $(a_0, a_1, \ldots, a_{N-1}) \in \mathbb{Z}^N$. Note that the product of two polynomials in R is simply the convolution product of their corresponding vectors

The obvious way to measure the size of an element $a \in \mathbb{Z}[X]/(X^N - 1)$ is by the Euclidean norm $(\sum a_i)^{1/2}$ of its vector of coefficients. However, when working with the ring R and its sublattices [8, 9], it is better to work with the centered norm, which is defined in the following way. Let $\mu_a = (1/N)\sum a_i$ denote the average of the coefficients of the polynomial $a(X) \in R$. Then the *centered norm* $\|a\|$ of a is defined by

$$\|a\|^2 = \sum_{i=0}^{N-1} (a_i - \mu_a)^2 = \sum_{i=0}^{N-1} a_i^2 - \frac{1}{N}\left(\sum_{i=0}^{N-1} a_i\right)^2. \tag{1}$$

For randomly chosen polynomials a and b, the norm is quasi-multiplicative,

$$\|a * b\| \approx \|a\| \cdot \|b\|.$$

When considering n-tuples of elements of R, there is no reason that they should be centered around the same value. In general we define the centered norm of an n-tuple $(a, b, \ldots, c)$ with $a, b, \ldots, c \in R$ by the formula

$$\|(a, b, \ldots, c)\|^2 = \|a\|^2 + \|b\|^2 + \cdots + \|c\|^2. \tag{2}$$

Remark 3. Notice that the centered norm is simply the Euclidean norm after projecting the vector of coefficients into the space orthogonal to $(1, 1, \ldots, 1)$. The reason that it is best to use the centered norm is because an attacker can always work with centered vectors himself, so there is nothing to be gained by using non-centered norms. Mathematically, the reduction to centered norms is reflected in the decomposition

$$\frac{\mathbb{Z}[X]}{(X^N - 1)} \overset{\sim}{=} \mathbb{Z} \times \qquad \frac{\mathbb{Z}[X]}{(X^{N-1} + X^{N-2} + \cdots + X + 1)},$$

which essentially allows the attacker to work in the second factor. This decomposition also illustrates why it is important that N should be prime, or at least should have no small prime factors, since if N factors, then there is a further decomposition of $\mathbb{Z}[X]/(X^N - 1)$. This observation underlies Gentry's attack [2] on the NTRU lattice when N is highly composite.

Remark 4. In applications a polynomial (or vector) a is only known modulo q, so some extra care must be taken to compute the centered norm. See Section 6.3 for more details.

The *Convolution Modular Lattice* L_h associated to the polynomial

$$h(X) = h_0 + h_1X + h_2X^2 + \cdots + h_{N-1}X^{N-1} \in R$$

is the set of vectors $(u, v) \in R \times R \cong \mathbb{Z}^{2N}$ satisfying

$$v(X) = h(X) * u(X) \pmod{q}.$$

It is easy to see that L_h is generated by the rows of the following matrix.

$$L_h = \text{RowSpan} \left(\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{N-1} \\ 0 & 1 & \cdots & 0 & h_{N-1} & h_0 & \cdots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & h_1 & h_2 & \cdots & h_0 \\ \hline 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array}\right)$$

We note that a convolution modular lattice has a rotational invariance property, since if $(u, v) \in L_h$, then

$$(X^i * u, X^i * v) \in L_h \quad \text{for all } 0 \le i < N.$$

Notice that each of the rotations $(X^i * u, X^i * v)$ has the same (centered) norm as (u, v). We write $R * (u, v)$ for the sublattice of L_h generated by (u, v) and all of its rotations, or equivalently for the R-submodule of R^2 generated by (u, v).

Micciancio [20] has observed that the matrix for L_h is given in Hermite Normal Form. Since any matrix can be put into Hermite Normal Form in polynomial time, it follows that the matrix for L_h conveys the minimum possible amount of information concerning short vectors in L_h.

If the polynomial h has a decomposition of the form

$$h \equiv f^{-1} * g \pmod{q}$$

with polynomials f and g having small coefficients (see below for the precise definition of small), then we say that L_h is an *NTRU Lattice* and denote it by L_h^{NT}.

Our goal is to directly relate both the NTRU Public Key Cryptosystem and the NTRU Signature Scheme to lattice problems in L_h^{NT}. To do this, we make the following definitions and assumptions:

(1) The security parameters N and q are related by

$$q = O(N).$$

In practice, we generally take $\frac{1}{3}N \le q \le \frac{2}{3}N$.

(2) A *small polynomial* is a polynomial whose coefficients are $O(1)$, that is, a polynomial whose coefficients are bounded independently of N. The (centered) norm of a small polynomial $a(X)$ satisfies

$$\|a\| = O(\sqrt{N}).$$

Remark 5. The *Gaussian heuristic* provides a method for predicting properties of a "random" lattice. We make a number of remarks concerning this heuristic which will be useful in our subsequent work. See [8] or [9] for details.

(1) The Gaussian heuristic predicts that the shortest vector in a "random" lattice L (of large dimension) has size approximately

$$\lambda_{\text{Gauss}}(L) = \sqrt{\dim(L)/2\pi e} \cdot \text{Det}(L)^{1/\dim(L)}.$$

Similarly, most closest vector problems for L have a solution whose size is approximately $\lambda_{\text{Gauss}}(L)$.

(2) A general convolution modular lattice L_h has dimension $2N$ and determinant q^N, so its probable shortest vector and closest vectors have size approximately

$$\lambda_{\text{Gauss}}(L_h) = \sqrt{Nq/\pi e} = O(N). \tag{3}$$

Notice that L_h contains N linearly independent vectors of length $q = O(N)$, namely the bottom N rows of its matrix. Small linear combinations of these "q-vectors" are the only obvious vectors of length $O(N)$ in L_h.

(3) If $(u, v) \in L_h$, then the vector obtained by reducing the coordinates of u and v modulo q is in L_h. Thus L_h contains a large number of vectors of length $O(q\sqrt{N}) = O(N^{3/2})$. Further, it is easy to find vectors in L_h whose distance to a given vector is at most $O(N^{3/2})$.

(4) In an NTRU lattice L_h^{NT}, the polynomial h has the form $h \equiv f^{-1} * g \mod q$ for small polynomials f and g, so L_h^{NT} contains the short vector (f, g). More generally, all of the rotations $(X^i * f, X^i * g)$ are short vectors in L_h^{NT} having length $O(\sqrt{N})$. Based on the Gaussian heuristic, these secret short vectors are probably $O(\sqrt{N})$ smaller than any vector not in the subspace $R * (f, g)$ that they span.

See Section 4.1 for some further asymptotics.

Definition 1. Let L_h^{NT} be an NTRU lattice. The *NTRU Lattice Key Problem* is to find a vector of length $O(\sqrt{N})$ in L_h^{NT}.

Remark 6. The NTRU lattice L_h^{NT} contains the subspace $R * (f, g)$ generated by vectors of length $O(\sqrt{N})$, so the NTRU Lattice Key Problem always has solutions. The Gaussian heuristic predicts that the shortest vector in L_h^{NT} that is linearly independent to the subspace $R * (f, g)$ has length $O(N)$. Hence it is highly probable that all solutions of the NTRU Lattice Key Problem are given by short multiples $(u * f, u * g)$, i.e., with $u \in R$ a polynomial of size $\|u\| = O(1)$.

3. NtruSign: Key Creation, Signing, and Verification

In this section we describe NtruSign key generation and the NtruSign signing and verification protocols. In the next section we will explain how an NtruSign signature solves an approximate CVP.

3.1. **Key generation.** Recall [8] that in an NtruEncrypt public/private key pair, the private key consists of two polynomials f and g having small coefficients and satisfying $\|f\| = \|g\| = O(\sqrt{N})$, and the public key is the polynomial h defined by the congruence

$$h \equiv f^{-1} * g \pmod{q}.$$

An NtruSign public/private key pair is created in exactly the same way: the key creator chooses (f, g) and forms $h \equiv f^{-1} * g \pmod{q}$. However, as part of his private key, the key creator also computes two additional polynomials F and G satisfying

$$f * G - g * F = q, \quad \text{and} \quad \|F\|, \|G\| = O(N).$$

We note that the rotations of (f, g) and (F, G) then form a basis for L_h, see Lemma 2 of Section 5.

In Section 5 we describe a method for efficiently computing an (F, G) pair for any given (f, g). More precisely, we show that if f and g are chosen to satisfy

$$\|f\| \approx c\sqrt{N} \quad \text{and} \quad \|g\| \approx c\sqrt{N} \tag{4}$$

for a given constant c, then it is possible to find an associated (F, G) satisfying

$$\|F\| \approx \|G\| \approx cN/\sqrt{12} \tag{5}$$

For the moment we will assume that the signer has such an (F, G) pair at his disposal.

3.2. **Signing.** To sign a digital document D, the signer first hashes D to produce a message digest $m = (m_1, m_2)$ composed of two random mod q polynomials m_1 and m_2. In section 4.4 we discuss in more detail the requirements on this hash function. For now, we assume that it is a randomized mapping that fulfills the necessary security requirements.

The signature on D is a vector $(s, t) \in L_h^{\mathrm{NT}}$ that is very close to m. The signer finds (s, t) by expressing (m_1, m_2) as a $\mathbb{Q}$-linear combination of his short basis vectors and then rounding the coefficients to the nearest integer. This standard method of approximately solving a CVP using a "good basis" of a lattice was already suggested for use in cryptography by [5].

Algorithmically, this procedure for the NTRU lattice can be described as follows:

- Compute polynomials $a, b, A, B \in \mathbb{Z}[X]/(X^N - 1)$ by the formulas

$$\begin{aligned} G * m_1 - F * m_2 &= A + q * B, \\ -g * m_1 + f * m_2 &= a + q * b, \end{aligned} \tag{6}$$

  where a and A are chosen to have coefficients between $-q/2$ and $q/2$.
- Compute polynomials s and t as

$$\begin{aligned} s &\equiv f * B + F * b \pmod{q}, \\ t &\equiv g * B + G * b \pmod{q}. \end{aligned} \tag{7}$$

The polynomial s is the signature on the digital document D for the public key h.

Remark 7. In practice, only b and B will be needed to create the signature, and only s is needed to form the signature. We also observe that (s, t) is in the NTRU lattice L_h^{NT}, since we can write

$$(s, t) = B * (f, g) + b * (F, G) \pmod{q}.$$

Remark 8. For a polynomial $P(X) \in \mathbb{Q}[X]$ with rational coefficients, we use the notation $\lfloor P \rceil$ to denote the polynomial obtained by rounding each coefficient of P to the nearest integer. Then the full signing process may be summarized by the following matrix equation, which shows that we are using our short basis $\{(f, g), (F, G)\}$ in the standard way to find approximate solutions to CVP:

$$\begin{aligned} \begin{pmatrix} s & t \end{pmatrix} = \begin{pmatrix} B & b \end{pmatrix} \begin{pmatrix} f & g \\ F & G \end{pmatrix} &= \left\lfloor \begin{pmatrix} m_1 & m_2 \end{pmatrix} \begin{pmatrix} G/q & -g/q \\ -F/q & f/q \end{pmatrix} \right\rceil \begin{pmatrix} f & g \\ F & G \end{pmatrix} \\ &= \left\lfloor \begin{pmatrix} m_1 & m_2 \end{pmatrix} \begin{pmatrix} f & g \\ F & G \end{pmatrix}^{-1} \right\rceil \begin{pmatrix} f & g \\ F & G \end{pmatrix} \end{aligned} \tag{8}$$

Remark 9. Note that a signature on a document D with message digest $m = (m_1, m_2)$ will also sign $m' = (m_1', m_2')$ as long as m' is sufficiently close to m. This forces some requirements on the choice of hash function which are discussed further in section 4.4. Interestingly, the signer can also take advantage of this feature in the following way. Rather than signing m, the signer perturbs m by a small amount to obtain m' and then signs m' instead. The signature thus generated will still be valid on m, while the perturbation makes it harder to extract useful information from a transcript of signatures. Further, the introduction of appropriately generated perturbations may make transcript analysis many orders-of-magnitude more difficult. See section 6.6 for some further details.

3.3. **Verification.** Let s be a putative NTRUSIGN signature for the message digest $m = (m_1, m_2)$ and public key h. The signature will be valid if it demonstrates that the signer knows a lattice point in L_h^{NT} that is sufficiently close to the message digest vector m. Verification thus consists of the following two steps:

- Compute the polynomial
$$t \equiv h * s \pmod{q}.$$
(Note that by definition, (s, t) is a point in the lattice L_h^{NT}.)
- Compute the (centered) distance from (s, t) to (m_1, m_2) and verify that it is smaller than a prespecified value `NormBound`.
In other words, check that
$$\big\|(s - m_1, t - m_2)\big\| \le \texttt{NormBound}. \tag{9}$$

3.4. **Why Verification Works.** A valid signature demonstrates that the signer knows a lattice point that is within `NormBound` of the message digest vector m. Clearly the smaller that `NormBound` is set, the more difficult it will be for a forger, without knowledge of the private key, to solve this problem. It is thus important to analyze how small we can set the bound `NormBound`, while still allowing valid signatures to be efficiently generated by the signer.

From (6) and (7) (or using (8)), we can calculate
$$(m_1, m_2) - (s, t) = \begin{pmatrix} A/q & a/q \end{pmatrix} \begin{pmatrix} f & g \\ F & G \end{pmatrix}.$$
We recall that the coefficients of a and A are between $-q/2$ and $q/2$, and hence
$$m_1 - s = \epsilon_1 * f + \epsilon_2 * F \quad \text{and} \quad m_2 - t = \epsilon_1 * g + \epsilon_2 * G, \tag{10}$$
where $\epsilon_1 = A/q$ and $\epsilon_2 = a/q$ are polynomials whose coefficients are between $-1/2$ and $1/2$.

As m_1 and m_2 vary across all mod q polynomials, it is easy to check that A varies uniformly across all mod q polynomials, so to all intents and purposes, the coefficients of ϵ_1 may be treated as independent random variables that are uniformly distributed in the interval $(-1/2, 1/2)$. Hence on average we have $\|\epsilon_1\| \approx \sqrt{N/12}$. A similar remark applies to a and ϵ_2, so also $\|\epsilon_2\| \approx \sqrt{N/12}$. (See also Remark 14.)

We can now estimate the distance from (s, t) to (m_1, m_2) using $\|\epsilon_1\| \approx \|\epsilon_2\| \approx \sqrt{N/12}$ and the quasimultiplicativity of the norm:
$$\|(m_1 - s, m_2 - t)\|^2 = \|(\epsilon_1 f + \epsilon_2 F, \epsilon_1 g + \epsilon_2 G)\|^2 \approx \frac{c^2 N^3}{72}\left(1 + \frac{12}{N}\right). \tag{11}$$

3.5. **A Sample Parameter Set for NTRUSIGN.** As we will see in subsequent sections, the following parameter set appears to require at least as much effort to break as does an RSA 1024 bit key, that is, roughly 10^{12} MIPS-years:

$$(N, q, c) = (251, 128, 0.45). \tag{12}$$

Her c is defined by equations (4) and (5). In practice, we choose f and g to be binary polynomials with f having 73 ones and 178 zeros, and with g having 71 ones and 180 zeros. (We need $f(1)$ and $g(1)$ to be relatively prime, or else we will not be able to extend (f, g) to a full basis of L_h^{NT}.)

The parameters (12) yield a bound of $46601 \approx 215.87^2$ on the right hand side of (11). This is the average expected distance. Thus it is quite feasible to find signatures that satisfy a norm bound in (9) of, say, `NormBound` $= 250$.

However, in practice it is not necessary to set such a stringent norm bound. We will see that neither lattice reduction nor other forgery methods are capable of creating signatures whose norm is much under 400, so `NormBound` can be set much higher, reducing the risk that validly generated signatures will fail verification. If `NormBound` ≤ 310, the strongest attack on NTRUSIGN is a direct lattice attack to recover the private key from the public key. This attack has been thoroughly studied for a number of years, since it is the same as the recovery of an NTRUENCRYPT private key from its public key. In practice, for $N = 251$, we take `NormBound` $= 300$.

3.6. **Prototype timing results.** We have functioning prototype implementations of the signing, verification, and key generation algorithms running in compiled C on an 800MHz Pentium III processor. With parameters $N = 251$, $q = 128$, $d_f = 73$, and $d_g = 71$ as described above, our program can sign approximately 2000 documents per second and can verify approximately 3000 documents per second. Key generation, which has not yet been fully optimized, takes approximately 1.5 seconds with iteration factor $B = 5000$ (see Remark 15) to generate a key pair, with the majority of the time used to compute a vector (F, G) with norm $\|(F, G)\| \approx 45$. We expect that the running time of the key generation algorithm can be significantly reduced.

4. Security Analysis of NTRUSIGN

It is clear that to forge an NTRUSIGN signature, a forger must be able to produce a lattice point sufficiently close to the message digest point, i.e., he must solve an approximate CVP problem in the NTRU lattice. In this section we consider three ways a forger might try to do this.

First, we consider the difficulty of producing a signature on a message directly from the public key. That is, we consider the difficulty of producing a pair $(s, t) \in L_h^{\mathrm{NT}}$ satisfying (9) without knowledge of f and g and without the availability of a transcript of valid signatures. Thus we examine how hard it is to solve CVP in these lattices within given approximation bounds. The only known approaches to solving this problem are:

(A) Lattice reduction techniques to locate a close lattice point.
(B) Exhaustive search, i.e., guess a lattice point and hope it is close, or guess a close point and hope it is a lattice point.
(C) Some combination of (A) and (B).

We analyze each of these in turn.

Second, we consider attacks that may be made possible by relations between message representatives. We demonstrate that a hash function that randomly maps the message into $[0,q)^N$ is sufficiently strong to make any such attack infeasible.

Finally, we consider the information that an attacker gains from a long transcript of signatures. Over time, this will leak information about the geometry of the lattice. Our analysis, summarized in this paper and presented in full in [7], demonstrates that although some additional information becomes available to an attacker after approximately 10,000 signatures, there are no effective transcript attacks known that require fewer than 10^8 signatures. In section 4.5 we summarize the general transcript analysis. In section 4.6 we provide more detail on the information that becomes available after 10,000 signatures and show that it does not appear to provide a realistic avenue of attack.

4.1. **Attacking NTRUSIGN Using Lattice Reduction.** An obvious way to use lattice reduction is to try to find a very short nonzero vector in L_h^{NT}, since (f,g) and its rotations are probably the shortest such vectors. With the parameter choice (12), this problem is identical to the problem of breaking an NTRUENCRYPT public key with the same parameters. Experiments give an estimated breaking time greater than 10^{12} MIPS years for the parameters (12).

Another way to use lattice reduction is to try to directly locate a valid signature (s,t). This problem is clearly an approximate closest vector problem (appr-CVP), since the signature (s,t) is required to be a lattice point that is close to a non-lattice point generated from the digital document via a hash function. It is generally accepted that for a "reasonably random" lattice, the difficulty of appr-CVP is measured by the dimension of the lattice and by the expected distance of a randomly chosen point in space to the closest point of the lattice. The Gaussian heuristic (3) suggests that the expected distance from a point in space to the closest point in L_h^{NT} is $\sqrt{Nq/\pi e}$. Thus a potential forger is presented with a random point in space and he must locate a point in L_h^{NT} whose distance to the given point is at most

$$\frac{\texttt{NormBound}}{\sqrt{Nq/\pi e}} \tag{13}$$

times the expected distance to the actual closest point of L_h^{NT}.

For the particular parameters described in (12), the Gaussian value is $\sqrt{Nq/\pi e} \approx 61.3$. Hence setting $\texttt{NormBound} = 300$ means that the forger needs to find a point that is no more than 4.89 times the expected shortest distance, while if $\texttt{NormBound} = 225$, then this ratio goes down to 3.67. Extensive experiments have shown that solving this approximate closest vector problem becomes more difficult as the ratio tends toward 1. In particular, for the suggested parameters (12), solving this approximate CVP is more difficult than breaking the public key (i.e., than finding (f,g) directly), even when the ratio is as high as 7.91, which corresponds to $\texttt{NormBound} = 485$.

Remark 10. In the NTRU lattices used in practice, one has $q = O(N)$, typically $\frac{1}{3}N \le q \le \frac{2}{3}N$. In this case, one easily sees that the norm bound must satisfy

$$\texttt{NormBound} = O(N^{3/2}).$$

Thus the signer is presented with a random target point in space and his task is to find a lattice point whose distance from the target point is at most

$$O\left(N^{3/2}\right) = O\left(\sqrt{2N} \cdot \sqrt{\frac{Nq}{\pi e}}\right).$$

Notice that this last expression is the square root of the dimension of the lattice multiplied by the expected shortest distance to a lattice vector (as predicted by the Gaussian heuristic (3)). Thus an NTRUSIGN signature solves appr-CVP up to a factor of $O(\sqrt{\dim L})$, but as we have seen, it does so with quite a small constant. Interestingly, as is shown in the next section, an attacker with no knowledge of the private key can also solve appr-CVP up to the same factor $O(\sqrt{\dim L})$, but with a much larger constant. The security of NTRUSIGN is based upon the fact that as the constant decreases, all known methods of solving appr-CVP (without knowledge of the private basis) become exponentially more difficult. This includes the use of exhaustive or collision searches, lattice reduction methods, or any combination thereof.

Remark 11. An important issue is whether the (approximate) shortest and closest vector problems in convolution modular lattices are as difficult as in more general classes of lattices. This is certainly an interesting question. The principal method for approximating shortest or closest vectors in a general lattice is the LLL algorithm [17] and its generalizations and enhancements such as [14, 15, 16, 24, 25, 27]. The evidence to date is that there are no special algorithms for convolution modular lattices that work significantly better than standard LLL-type lattice reduction algorithms (although see [18] for work in this area), and lattice reduction algorithms do not appear to work better on convolution modular lattices than they do on general lattices.

4.2. **Attacking NTRUSIGN Using Exhaustive Search.** To analyze the difficulty of using an exhaustive search to forge an NTRUSIGN signature, we first observe that an integer point chosen at random in space has only a q^{-N} probability of being in L_h. A more effective method for attempting to solve appr-CVP in a convolution modular lattice L_h, and thus to forge an NTRUSIGN signature, was noted independently by Craig Gentry, Jakob Jonsson and Jacques Stern (see [3]). They observe that an attacker may preselect half of the coefficients of s and t to have any desired values and then solve

$$t \equiv h * s \pmod{q}$$

for the remaining N coefficients. This creates a point (s, t) in the lattice L_h, half of whose coordinates are freely chosen by the attacker. Of course, the other N coordinates will generally be randomly and uniformly distributed modulo q.

Thus the attacker can easily create a lattice point (s, t) so that half of the coordinates of the difference $(m_1, m_2) - (s, t)$ vanish, and the other half are independently uniformly distributed modulo q. The probability of success of this approach is thus measured by the probability that the sum of the squares of N integers chosen randomly and uniformly from the interval $[-q/2, q/2]$ is less than $\texttt{NormBound}^2$. It is

surprisingly hard to obtain a precise estimate for this probability.[1] The following elementary upper bound will suffice for our purposes.

Proposition 1. *Let $\mathcal{Q}$ be a finite set of real numbers, let $q = \#\mathcal{Q}$, let $X_1, \dots, X_N$ be independent random variables that are uniformly distributed on $\mathcal{Q}$, and let $Y = \sqrt{X_1^2 + \cdots + X_N^2}$. Then for all $A > 0$ and all $t > 0$,*

$$\text{Prob}(Y \le A) \le e^{A^2 t} \left(\frac{1}{q} \sum_{x \in \mathcal{Q}} e^{-x^2 t} \right)^N .$$

In particular,

$$\text{Prob}(Y \le A) \le \left(\frac{\sqrt{e}}{q} \sum_{x \in \mathcal{Q}} e^{-x^2 N/2A^2} \right)^N .$$

Proof. We estimate the number of points in the sphere $\|\mathbf{x}\| \le A$ that have coordinates chosen from the set $\mathcal{Q}$. Thus

$$\begin{aligned} \#\{\mathbf{x} \in \mathcal{Q}^N : \|\mathbf{x}\| \le A\} &\le \sum_{\mathbf{x} \in \mathcal{Q}^N} e^{(A^2 - |\mathbf{x}\|^2)t} \quad \text{for all } t > 0, \\ &= e^{A^2 t} \left(\sum_{x \in \mathcal{Q}} e^{-x^2 t} \right)^N . \end{aligned}$$

Since the coordinates of $\mathbf{x}$ are chosen independently and uniformly in the set $\mathcal{Q}$, we have

$$\text{Prob}(\|\mathbf{x}\| \le A) = \frac{\#\{\mathbf{x} \in \mathcal{Q}^N : \|\mathbf{x}\| \le A\}}{\#\mathcal{Q}^N},$$

which completes the proof of the first part of the proposition.

The second estimate follows from the first estimate by setting $t = N/2A^2$. □

We apply the proposition with the parameters

$$N = 251, \quad q = 128, \quad \mathcal{Q} = \{-64, -63, \dots, 63\}.$$

The results are listed in Table 1. It is clear from examining the table that even if `NormBound` is chosen as large as 380, the chances of a successful forgery by this method are negligable.

Remark 12. The analysis given in Proposition 1 is only an approximation, because it uses the standard norm, rather than the centered norm. However, this makes only a small difference, because for the vast majority of random mod q vectors, the average value of the coefficients will be very small. A rigorous mathematical analysis is fairly complicated, so we use the following experiment to illustrate. We computed 10000 random mod q vectors with $N = 251$ and $q = 128$, and for each vector we computed the standard norm and the centered norm. Table 2 shows that there is only a small difference. Since Proposition 1 is an upper bound and since we have chosen parameters so that this upper bound is far smaller than is necessary, the small effect of using centered norms instead of standard norms does not affect our security estimates.

[1] One might think that this problem could be easily solved by esimating volumes. The difficulty is that one must compute the volume of the intersection of an N-sphere of radius R with an N-cube of side $2B$, where $B < R < B\sqrt{N}$. Thus some parts of the sphere stick out of the sides of

A	Upper Bound for Prob($\|\mathbf{x}\| \leq A$)
300	$2^{-178.44}$
310	$2^{-166.69}$
320	$2^{-155.36}$
350	$2^{-123.72}$
380	$2^{-95.35}$
400	$2^{-78.10}$
420	$2^{-62.09}$
480	$2^{-20.76}$

TABLE 1. Upper Bound for Forgery Probability — $N = 251$, $q = 128$

	Minimum	Mean	Maximum	Standard Deviation
Standard Norm	515.013	585.478	642.626	16.5245
Centered Norm	513.174	584.244	640.476	16.5747

TABLE 2. Comparison of Standard Norm and Centered Norm

Remark 13. The upper bound in Proposition 1 can be slightly improved by applying the functional equation of the classical theta function. The result is as follows, we omit the proof.

$$\text{Prob}(\|\mathbf{x}\| \leq A) < \frac{\pi^{N/2}}{\Gamma(1+N/2)} \cdot \left(\frac{A}{q}\right)^N (1 + o(1)). \tag{14}$$

4.3. **Attacking NTRUSIGN by Combining Lattice Reduction with Exhaustive Search.** There remains the possiblity of combining the two methods. Thus a forger could preselect somewhat fewer than N coordinates and use lattice reduction techniques on a lower dimensional lattice to find the remaining coordinates. Thus suppose that a forger preselects αN coordinates of s and t for some choice of $0 \leq \alpha \leq 1$. He then uses lattice reduction techniques on a lattice of dimension $(2-\alpha)N$ and determinant $q^{N(1+\alpha)}$ to make the remaining $(1-\alpha)N$ coordinates as small as possible. Notice that $\alpha = 0$ corresponds to pure lattice reduction and $\alpha = 1$ corresponds to pure exhaustive search. As α increases, the fundamental ratio (cf. (13))

$$\frac{\texttt{NormBound}}{\sqrt{\dfrac{(2-\alpha)N}{2\pi e}} \cdot q^{(1+\alpha)/(2-\alpha)}}$$

decreases, and when it passes below 1, the Gaussian heuristic says that it is very unlikely for any solutions to exist. Table 3 gives the largest allowable value of α and the corresponding lattice dimension for various values of `NormBound`. For example, `NormBound` = 300 gives a value of $\alpha = 0.3835$, which corresponds to a lattice of dimension 407. Thus a lattice reduction attack cannot hope to be reduced below dimension 407 (down from 502). Further, as the dimension is reduced towards 407,

the cube. Fortunately, for our applications it suffices to use the entire volume of the sphere as an upper bound for the volume of the intersection.

NormBound	α	Lattice Dim
300	0.3772	407
310	0.3835	405
320	0.3895	404
350	0.4062	400
380	0.4213	396
400	0.4305	393
420	0.4392	391
480	0.4624	385

TABLE 3. Minimum Usable Lattice Dimension — $N = 251$, $q = 128$

the advantage gained from the reduction in dimension is at least partially eliminated due to the decrease in the Gauss ratio.

4.4. **Hash Function Requirements.** The hashing of the document D to produce a message digest $m = (m_1, m_2)$ is actually a two stage process. First a standard secure hash function H_1 is applied to D to give an output $H_1(D)$ consisting of β bits for an appropriate choice of β. (Typical choices would be $\beta = 160$ or $\beta = 256$. In any case, β should be sufficiently large so as to make it infeasible to search a set containing $2^{\beta/2}$ elements.) Next a (public) function

$$H_2 : (\mathbb{Z}/2\mathbb{Z})^\beta \longrightarrow (\mathbb{Z}/q\mathbb{Z})^{2N}$$

is applied to $H_1(D)$ to yield the message digest $m = H_2(H_1(D))$. The function H_2 should map the 2^β possible H_1 hash values in a reasonably uniform manner into the set of q^{2N} possible message digests.

One point that must be examined carefully is the possibility that two points m and m' in the image of $H = H_2 \circ H_1$ might be very close together. Two potential attacks arise here.

First, if a very close pair exists and if the signer can be induced to sign the corresponding digital documents D and D', then there is a small, but nontrivial, possiblity that the difference of the signatures $s - s'$ is a small element of the underlying lattice, which might reveal the private key. In practice, experiments have shown [23] that if m and m' differ by one bit, there is a 5% chance that $s - s'$ is a rotation of the private polynomial f. Thus the existence of documents D and D' having small difference $m - m'$ could endanger all NTRUSIGN implementations using a common mapping H. We refer to this as a "key recovery attack", because it enables the attacker to recover the private key.

Second, if m and m' are close together, but not so close that they endanger the private key, then there is still a chance that a valid signature s on m will also be a valid signature on m'. This is true because

$$\|s - m'\| \approx \sqrt{\|s - m\|^2 + \|m - m'\|^2}.$$

An attacker who can obtain the signature of m can then fraudulently present it as the signature of m'. We refer to this as a "collision attack", because it corresponds to attacks based on finding collisions in hash functions.

We now consider each attack in turn and explain why it is highly unlikely that there is even a single useful (to the attacker) pair of documents D and D', and thus why neither potential attack is of practical significance.

First, let us consider then the probability that two points m and m' are sufficiently close to each other to enable the key recovery attack. To simplify computations (and aid the attacker), we make the assumption that $m_1 = 0$, so m has the form $m = (0, m_2)$. Thus the map H_2 has the form

$$H_2 : (\mathbb{Z}/2\mathbb{Z})^\beta \longrightarrow (\mathbb{Z}/q\mathbb{Z})^N.$$

We identify $(\mathbb{Z}/q\mathbb{Z})^N$ with the set of N-tuples

$$(\mathbb{Z}/q\mathbb{Z})^N = \big\{(a_1, \ldots, a_N) \in \mathbb{Z}^N : -q/2 < a_i \le q/2\big\} \subset \mathbb{R}^N,$$

and we will suppose first that the mapping H_2 is close to uniform with respect to the usual measure on $\mathbb{R}^N$. For $u, u' \in (\mathbb{Z}/2\mathbb{Z})^\beta$, we will use the notation $m = H_2(u)$ and $m' = H_2(u')$. We compute

$$\begin{aligned}
\#\big\{(u,u') \in (\mathbb{Z}/2\mathbb{Z})^\beta : \|m - m'\| \le B\big\} &\le 2^{2\beta} \cdot \mathrm{Prob}\big(\|m - m'\| \le B\big) \\
&\le 2^{2\beta} \cdot \frac{\text{Volume of an } N\text{-ball of radius } B}{\text{Volume of an } N\text{-box of side } q} \\
&\approx 4^\beta \frac{\pi^{N/2}}{\Gamma(1+N/2)} \left(\frac{B}{q}\right)^N.
\end{aligned}$$

For example, taking $N = 251$ and $q = 128$ as usual, we find that when $\beta = 160$ and (coincidentally) $B = 160$, the expected number of bad pairs (u, u') is less than 2^{-81}. In other words, if H_2 is reasonably uniform, there is virtually no chance that there are any pairs sufficiently bad to endanger the private key.

Second, we consider the resistance of a random mapping H_2 to collision attacks. The measure of this resistance is the number of points that need to be generated by this mapping before there is a chance of greater than 50% that two image points exist within a distance B_{coll} of each other. This can be done by a similar easy calculation as follows. Consider where a mapping H_2 can place a succession of points. The first point can go anywhere. So long as the second point is at least B_{coll} from the first point, there is no collision. The third point needs to avoid (at worst) two balls of radius B_{coll}. Continuing with this reasoning, we find:

$$\begin{aligned}
&\mathrm{Prob}(\text{no collision on } (n+1)\text{st random point}) \\
&\quad\le \prod_{k=0}^{n} \left(1 - k \cdot \frac{\text{Volume of an } N\text{-ball of radius } B_{\mathrm{coll}}}{\text{Volume of an } N\text{-box of side } q}\right) \\
&\quad= \prod_{k=0}^{n} (1 - k \cdot C), \qquad \text{where } C = \frac{\pi^{N/2}}{\Gamma(1+N/2)} \left(\frac{B_{\mathrm{coll}}}{q}\right)^N, \\
&\quad\approx 1 - \sum_{k=0}^{n} k \cdot C, \qquad \text{assuming that } C \ll n, \\
&\quad\approx 1 - \frac{C \cdot n^2}{2}.
\end{aligned}$$

In order for the probability of collision to be $\frac{1}{2}$, therefore, we have the familiar birthday paradox formula:

$$n_{\text{coll}} \geq \sqrt{1/C} = \sqrt{\frac{\Gamma(1+N/2)}{\pi^{N/2}}} \left(\frac{q}{B_{\text{coll}}}\right)^{N/2}.$$

For example, taking $(N, q, \texttt{NormBound}) = (251, 128, 300)$, we find:

$$\begin{aligned} \text{Avg}(B_{coll}) &= \sqrt{\texttt{NormBound}^2 - \text{Avg}(\|s-m\|)^2} \\ &= \sqrt{300^2 - 215.87^2} \\ &= 208.32, \end{aligned}$$

giving

$$n_{\text{coll}}(H_2, N, q, \texttt{NormBound}) \geq 2^{152}.$$

If we instantiate H_1 with a standard hash function such as SHA-1, $n_{\text{coll}}(H_1) = 2^{80}$. The collision-resistance of a random H_2 is therefore more than suitable for the expected security of the system.

In summary, the probability that a given public mapping contains in its image two points $m = H_2(u)$ and $m' = H_2(u')$ that are close enough to one another to risk compromising the private key is extremely small. Further, even if our aim is simply to prevent collisions that allow a signature on one message to be presented as a signature on another, a random mapping into $[0, q-1]^N$ is sufficiently secure for our purposes.

Finally, we note that the proposed attacks assume that all signers are using the same public function H_2. The potential gain to an attacker from inverting H_2 can be further reduced by appending the public key of the signer to the digital document before calling the function H_1 and/or including the public key as an additional argument in the public function H_2.

4.5. **Transcript Analysis.** In this section we analyze what information an attacker gets from each signed message. Recall that the difference between the signature vector and the message digest vector is a short vector of norm at most `NormBound`. The signature vector is a lattice vector, while the message digest vector is a random point in $\mathbb{Z}^{2N}$ mod q. We view a transcript as the list of differences $(m_1, m_2) - (s, t)$, each of which is a short vector.

(1) Since we are assuming that the message digest is created using a hash function that behaves as a random oracle, an active attacker (one who asks for signatures for chosen messages) will have no more information than a passive one (one who only analyzes a transcript of signed messages).
(2) If the signature difference vectors were uniformly distributed on (or in) a sphere of radius `NormBound`, then a transcript would yield no useful information for solving appr-CVP.

The second point is a very strong statement and deserves some justification. Essentially it is true because an attacker can himself create a transcript of this form. He simply chooses random lattice points $a_1, a_2, \ldots$, chooses random vectors $r_1, r_2, \ldots$ of length equal to (or less than) `NormBound`, and takes as his transcript the "signatures" $a_1, a_2, \ldots$ on the "message digests" $a_1 + r_1, a_2 + r_2, \ldots$.

Thus under the assumption (2), if an efficient algorithm exists for solving appr-CVP using a transcript, then an efficient algorithm exists for solving appr-CVP that does not require a transcript (which is believed not to be true).

Taking the above two points into consideration means that the only transcript information that an attacker can attempt to exploit is due to the fact that our signature differences are not randomly distributed on (or within) a ball of radius `NormBound`.

As we noted in Section 3.4, the method that we are using to solve appr-CVP using our good basis (f, g) and (F, G) leads to the equation

$$(m_1, m_2) - (s, t) = (A/q, a/q) \begin{pmatrix} f & g \\ F & G \end{pmatrix}, \tag{15}$$

where A and a are mod q vectors with coordinates chosen between $-q/2$ and $q/2$. More precisely, they are determined by the congruences

$$\begin{aligned} A &\equiv G * m_1 - F * m_2 \pmod{q}, \\ a &\equiv -g * m_1 + f * m_2 \pmod{q}. \end{aligned} \tag{16}$$

It is clear from (16) that as m_1 and m_2 vary over all mod q vectors, the value of A will vary uniformly over all mod q vectors, and similarly for a.

Remark 14. It is not true that a and A are independent of one another. In fact, it is easy to check that

$$f * A \equiv -F * a \pmod{q}, \tag{17}$$

So for a given (f, F), we see that a determines A, and vice versa. This is not a security issue, since the congruence (17) is simply a restatement of the fact that the formula for $m_1 - s$ in (15) has integer coefficients.

The information that an attacker can obtain from a transcript is a collection of polynomials

$$s - m_1 = \frac{A}{q} * f + \frac{a}{q} * F \qquad \textit{not reduced modulo } q,$$

where the coefficients of A and a are random mod q polynomials subject to the relation (17). (This is the information from s; similar information about g and G is obtained from t.) Thus a transcript may be viewed as a collection of polynomials

$$\epsilon_1 * f + \epsilon_2 * F,$$

where the coefficients of ϵ_1 and ϵ_2 are more-or-less randomly distributed in the interval $[-1/2, 1/2]$. Equivalently, a transcript is a random collection of points in a (centered) fundamental domain for the NTRU lattice spanned by the basis vectors (f, g) and (F, G).

Taking a simple average of a collection of signatures is not useful, since the average will be zero (or some other simple expression that reveals no useful information). However, there are other ways to take averages that introduce higher moments. The subject of moment averaging attacks was discussed in [9, 11, 12], and it is natural to ask what success similar attacks would have in this context.

A useful tool in studying moments is the notion of the reversal $\bar{c}(X) := c(X^{-1})$ of a polynomial $c(X)$. More precisely, the product of a polynomial with its reversal

will often have a nontrivial average. We denote this product by

$$\hat{c}(X) = c(X) * c(X^{-1}) = \sum_{k=0}^{N-1} \left(\sum_{i=0}^{N-1} c_i c_{i+k} \right) X^k.$$

Since the coefficients of $\hat{c}(X)$ involve products $c_i c_{i+k}$, the polynomal $\hat{c}(X)$ is a *second moment polynomial*. Similarly, its square $\hat{c}(X)^2$ is a *fourth moment polynomial*.

The first piece of information available to an attacker from a long transcript of signatures $s(X)$ is the average of the second moment polynomials $\widehat{s - m_1}$. Notice that these second moment polynomials are equal to

$$\begin{aligned} \widehat{s - m_1} &= (\epsilon_1 * f + \epsilon_2 * F) * (\bar{\epsilon}_1 * \bar{f} + \bar{\epsilon}_2 * \bar{F}) \\ &= \hat{\epsilon_1} * \hat{f} + \hat{\epsilon_2} * \hat{F} + \epsilon_1 * \bar{\epsilon_2} * f * \bar{F} + \epsilon_2 * \bar{\epsilon_1} * \bar{f} * F. \end{aligned}$$

As the number of signatures in the transcript goes to infinity, it turns out that $\hat{\epsilon_1}$ and $\hat{\epsilon_2}$ (essentially) approach constants. Further, and somewhat curiously, the cross terms in this expression (essentially) average out to zero, despite the fact that ϵ_1 and ϵ_2 are not independent. Hence by averaging the second moment polynomials over a sufficiently long transcript, an attacker may be able to recover the quantity

$$\hat{f} + \hat{F} = f * \bar{f} + F * \bar{F}. \tag{18}$$

Experiments indicate that in order to reconstruct this value, using either direct or lattice assisted averaging, it is necessary to compile a transcript consisting of 10,000 signatures, and possibly substantially more.

However, the quantity (18) does not appear to provide sufficient information to either obtain the private key or to forge a signature. This is because (18) combines information about the two quantities f and F, and there is no known way to directly untangle them. See Section 4.6 for some further comments on this problem.

Thus an attacker needs to go further and use fourth moment polynomials. As observed by Coppersmith in the case of [11] and independently noted by Gentry and Szydlo for NTRUSIGN [4], a transcript long enough to yield accurate limiting averages of fourth power moments should contain enough information to compromise the private key. In fact, the limiting value of an average of $\widehat{(s - m_1)}^2$ is a very complicated expression involving a linear combination of the three quantities $\hat{f}^2$, $\hat{F}^2$, and $\hat{f} * \hat{F}$. If this limiting value can be determined sufficiently accurately, then it can be combined with the value of (18) to separate and recover $\hat{f}$ and $\hat{F}$. Finally, the attacker would apply a method of Gentry and Szydlo [4] to $\hat{f}$ and $\hat{f} * h$ to recover f in polynomial time.

Note that in order for this attack to proceed, it is necessary to obtain the limiting average value of $\widehat{(s - m_1)}^2$ quite closely. And it is clear that averages of these fourth moment polynomials will converge quite slowly. In [11] it was remarked that a transcript length of 100 million was certainly far too short. More extensive experiments and a more refined analysis, detailed in [7], has confirmed that a practical attack would require more than 100 million signatures. (There are potentially some further problems to overcome having to do with taking of square roots; see [7] for details. If an attacker cannot take the correct square root, the transcript length required appears to be considerably longer even than this.)

We would like to thank Craig Gentry and Mike Szydlo for pointing out that the fourth moment analysis needed to be extended and included in the present

discussion. They also suggested some possible variations on second moment attacks by restricting to subtranscripts where the norms (or coefficients) of signatures meet certain boundary conditions. We have investigated this approach; it appears to have worse characteristics than fourth moment attacks, and thus to require transcripts of similar length.

4.6. **Gram matrices.** Recall (18) that the quantity

$$\hat{f} + \hat{F} = f * \bar{f} + F * \bar{F}$$

can be obtained from a successful second moment attack on an unperturbed transcript. (See section 6.6 for remarks on perturbations.) The analogous expression built from (g, G) can also be obtained, and after multiplication by h and reduction modulo q the cross terms $g * \bar{f} + G * \bar{F}$ and $\bar{g} * f + \bar{G} * F$ can be obtained. Let extra variables k, K be defined by $g = f * h + kq$ and $G = F * h + Kq$. The private information available only to the signer can be summarized in two secret matrices:

$$S = \begin{pmatrix} f & g \\ F & G \end{pmatrix}, \quad U = \begin{pmatrix} K & -k \\ -F & f \end{pmatrix},$$

where $f * G - g * F = q$ and $K * f - k * F = 1$.

For a matrix A with coefficients in R, let $\bar{A}^t$ denote the conjugate transpose of A. Then the public information after a successful second moment attack is given by

$$\bar{S}^t S = M = \begin{pmatrix} f * \bar{f} + F * \bar{F} & g * \bar{f} + G * \bar{F} \\ \bar{g} * f + \bar{G} * F & g * \bar{g} + G * \bar{G} \end{pmatrix}$$

and by

$$U \bar{U}^t = H M^{-1} \bar{H}^t,$$

where H is the matrix corresponding to the public key:

$$H = \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}.$$

Note that $S\bar{S}^t$ and $\bar{U}^t U$ are *not* revealed.

Let the Gram matrix corresponding to A be defined as $\mathrm{Gram}(A) = A\bar{A}^t$. The question of extracting information available from a successful second moment attack thus translates into the following problem, which we call the Gram Factorization problem:

Let A be as above. How difficult is it to recover A from knowledge of its Gram matrix $\mathrm{Gram}(A) = A\bar{A}^t$?

One way of looking at this problem is to view it as a generalization of the problem solved in [4]. In that paper, knowledge of a basis for an ideal (f) generated by a secret f and knowledge of $f * \bar{f}$ is shown to be sufficient to recover f (up to rotation and sign) in polynomial time. Here $H = US$ would play the part of (f) and $M = S^t S$ would play the part of $f * \bar{f}$. Thus the problem has a similar form. However, the technique of [4] relies heavily on the fact that polynomials in R have representations as N by N circulant matrices, and these matrices commute with eachother. The corresponding objects here are non-commuting $2N$ by $2N$ matrices. This appears to present a significant obstacle to generalizing the method of [4] to this context.

We can also look directly at the system of equations that must be solved in integers to recover the secret information. Let $a, b, c, d \in R$ satisfy

$$a * d - b * c = 1. \tag{19}$$

Suppose that the quantities

$$a * \bar{a} + b * \bar{b}, \qquad a * \bar{c} + b * \bar{d}, \qquad \bar{a} * c + \bar{b} * d, \qquad c * \bar{c} + d * \bar{d} \tag{20}$$

are known. How difficult is it to find (a, b, c, d)?

A first observation is that the relation (19) and the four quantities (20) are not independent. An obvious relation is

$$a * \bar{c} + b * \bar{d} = \overline{\bar{a} * c + \bar{b} * d}$$

so (20) really only gives three values. A less obvious relation is given by the formula

$$(a * \bar{a} + b * \bar{b})(c * \bar{c} + d * \bar{d}) = (a * d - b * c)(\overline{a * d - b * c}) + (a * \bar{c} + b * \bar{d})(\bar{a} * c + \bar{b} * d).$$

Thus (19) and (20) really only contain two independent pieces of information and we can rephrase the problem as follows.

Let $r, s \in R$ and suppose that the simultaneous equations

$$x * \tilde{z} + y * \tilde{w} = r, \qquad z * \tilde{z} + w * \tilde{w} = s, \qquad x * w - y * z = 1 \tag{21}$$

have a solution $(x, y, z, w) = (a, b, c, d) \in R^4$. How difficult is it to find a solution?

If we embed the ring R into the larger ring

$$R_{\mathbb{R}} = \frac{\mathbb{R}[X]}{(X^N - 1)}$$

and identify $R_{\mathbb{R}}^4$ with $\mathbb{R}^{4N}$, then the equations (21) for the Gram Factorization Problem describe a certain subvariety of $\mathbb{R}^{4N}$. The Gram Factorization Problem is then equivalent to finding a point in this variety having integer coordinates. Thus it appears that any method of solving these equations must rely heavily on number theoretic properties of R.

For each fixed pair $(r, s) \in R_{\mathbb{R}}$, the three equations (21) define an affine variety in $\mathbb{R}^{4N}$. We denote this variety by V, that is,

$$V = \{(x, y, z, w) \in \mathbb{R}^{4N} : x * \tilde{z} + y * \tilde{w} = r, \quad z * \tilde{z} + w * \tilde{w} = s, \quad x * w - y * z = 1\}.$$

The equations defining V appear to be nonlinear, but we can find a linear relation by eliminating one of the variables, for example by eliminating x. To do this, we use the identity

$$(z * \tilde{z} + w * \tilde{w}) * y - (x * \tilde{z} + y * \tilde{w}) * w + \tilde{z} * (x * w - y * z) = 0$$

and substitute in the values given by (21) to obtain

$$s * y - r * w + \tilde{z} = 0. \tag{22}$$

This says that the variety V lies in the plane defined by the linear equation (22), but of course it is not equal to that entire plane. What we find is that V can be defined by the equations

$$V = \{(y, z, w) \in R_{\mathbb{R}}^3 : s * y - r * w + \tilde{z} = 0, \quad z * \tilde{z} * r + w * \tilde{w} = 1\}.$$

The first equation shows that V lies in a vector subspace of dimension $2N$ inside the vector space $\mathbb{R}^{2N}$. The second equation is also quite interesting, but it is highly nonlinear.

Using the linear relation (22), we can formulate an SVP problem that will (probably) solve Gram Factorization. Suppose that $r, s \in R$ are given. We consider the lattice L_{rs} generated by the rows of the following matrix, where as usual we can either work over R or we can express everything in terms of circulant matrices and work over $\mathbb{Z}$.

$$L_{rs} = \begin{pmatrix} r & 0 & 1 \\ s & 1 & 0 \end{pmatrix}$$

The lattice L_{rs} has dimension $2N$. Its determinant is given roughly by

$$\mathrm{Det}(L_{rs}) \approx \sqrt{\det(r)^2 + \det(s)^2 + \det(r-s)^2}.$$

Here $\det(r)$ means the determinant of the circulant matrix associated to r, or equivalently, the absolute value of the image of r under the map

$$R \longrightarrow \mathbb{Z}, \qquad a(X) \longmapsto \prod_{i=0}^{N-1} a(\zeta^i).$$

Let $(x, y, z, w) = (a, b, c, d) \in R^4$ be the solution to the Gram Factorization Problem that we are seeking. Multiplying the matrix of L_{rs} on the left by (d, b), we see that the lattice L_{rs} contains the vector

$$\mathbf{t} = (\tilde{c}, -b, d) \in L_{rs}.$$

We will call $\mathbf{t}$ the target vector. The length of $\mathbf{t}$ satisfies

$$\frac{\|\mathbf{t}\|}{\lambda_{\mathrm{Gauss}}(L_{rs}))} \approx O\left(\frac{1}{\sqrt{N}}\right),$$

where $\lambda_{\mathrm{Gauss}}(L_{rs})$ denotes, as before, the length of the expected shortest vector in L_{rs} as predicted by the Gaussian heuristic. Thus we see that the target vector is probably the shortest vector in the lattice. The factor of $1/\sqrt{N}$ means that $\mathbf{t}$ bears roughly the same relationship to the length of the expected shortest vector in L_{rs} as the target does in the usual NTRU lattice of dimension $2N$.

We have performed experiments using LLL to extrapolate the block size and length of time necessary for lattice reduction techniques to locate the target $\mathbf{t}$. The lattice L_{rs} appears to be more difficult to reduce than the usual NTRU lattice, which is reasonable as the constant in the $O\left(1/\sqrt{N}\right)$ relation is rather large. The predicted breaking time with this approach significantly exeeds 10^{12} MIPS years.

It should be noted that the attacker has a further piece of information. He knows that within the $2N$-dimensional lattice L_{rs}, the sought for short vector lies on the subset defined by the nonlinear equation

$$z\tilde{z} + w\tilde{w} = 1. \tag{23}$$

(In terms of the coordinates, this is really N nonlinear equations in the $2N$ variables $(z_0, \ldots, z_{N-1}, w_0, \ldots, w_{N-1})$. Thus in principle, he only needs to work on the N-dimensional variety given by the intersection of the lattice hyperplane and the set (23). However, it is generally considered to be a very difficult problem to find integer points on nonlinear sets of high dimension, and in particular, there is no known way to take signficant advantage of nonlinear information in lattice reduction algorithms such as LLL.

5. A Sample Method for Generating NTRUSign keys

In this section we show how to generate the keys necessary for the signing algorithm, i.e., how to find two vectors whose rotations form a basis for the NTRU lattice. It is important to note that we will describe only one of several ways to find such vectors. It is a current research problem to make this process as efficient as possible.

In the following we consider 2×2 matrices with entries in the ring $\mathbb{Z}[X]$ or $\mathbb{Z}[X]/(X^N - 1)$, where it will be clear from context which ring we are using. The determinant of such a matrix is then an element of $\mathbb{Z}[X]$ or $\mathbb{Z}[X]/(X^N - 1)$. Our goal is to take a given top row for the matrix and to find a bottom row so that the determinant is equal to q.

To achieve this goal, we suppose that we are given two polynomials f and g. For example, f and g might be binary polynomials having d_f and d_g coefficients equal to one, respectively, and the remaining coefficients equal to zero. Let $F_1, G_1 \in \mathbb{Z}[X]$ be such that the matrix

$$M_1 = \begin{pmatrix} f & g \\ F_1 & G_1 \end{pmatrix} \tag{24}$$

is unimodular over $\mathbb{Z}[X]/(X^N - 1)$, i.e.,

$$\det(M_1) = f * G_1 - g * F_1 \equiv 1 \pmod{X^N - 1}. \tag{25}$$

We will use resultants to find such an F_1 and G_1, but we note that this is certainly not the only way, nor even the most efficient way, to find F_1 and G_1.

We begin by using standard methods to find polynomials $u, v \in \mathbb{Z}[X]$ satisfying

$$f * v + k_1 * (X^N - 1) = R_f, \tag{26}$$

$$g * u + k_2 * (X^N - 1) = R_g, \tag{27}$$

where R_f and R_g are the (integer) resultants of $(f, X^N - 1)$ and $(g, X^N - 1)$ respectively. The resultant of f can be straightforwardly calculated as $\prod_{i=1}^{N-1} f(X^i)$ modulo the cyclotomic polynomial $\phi(N) = 1 + x + x^2 + ... + x^{N-1}$.

Assuming that the resultants R_f and R_g are coprime, we then apply the (integer) extended Euclidean algorithm to obtain integers α and β satisfying

$$\alpha R_f + \beta R_g = 1.$$

Combining these relations gives the formula

$$(\alpha v) * f + (\beta u) * g = 1 \pmod{X^N - 1}.$$

Hence as long as $\gcd(R_f, R_g) = 1$, we have found polynomials $F_1 = -\beta u$ and $G_1 = \alpha v$ so that the matrix M_1 defined by (24) satisfies the unimodularity condition (25).

If the resultants are not coprime, then one can choose different f and g. It is also possible to rapidly perform initial tests on f and g to check if it is likely that their resultants might not be coprime. One important observation is that $f(1)$ will divide R_f and $g(1)$ will divide R_g, so a necessary condition for $\gcd(R_f, R_g) = 1$ is that

$$\gcd(f(1), g(1)) = 1.$$

We also observe that if N is prime and if p is a prime satisfying $p \equiv 1 \pmod{N}$, then

$$\text{Prob}(p \text{ divides } R_f) \approx 1 - \left(1 - \frac{1}{p}\right)^N.$$

In particular, if $p = 2N + 1$ is prime, then R_f and R_g each have approximately a 39.3% chance of being divisible by p, so it may be worthwhile to check that at least one of them is not divisible by $2N + 1$ before proceeding with the rest of the calculation.

Once we have a unimodular matrix (24), it is a trivial matter to get any determinant we want, since we may multiply the second row by a constant. Thus to produce a matrix with determinant q, we set

$$F_q = qF_1 \qquad \text{and} \qquad G_q = qG_1$$

and form the matrix

$$M_q = \begin{pmatrix} f & g \\ F_q & G_q \end{pmatrix}. \tag{28}$$

Then

$$\det(M_q) = f * G_q - g * F_q = q. \tag{29}$$

Proposition 2. *The rows of the matrix M_q defined by (28) generate the NTRU lattice L_h associated to h, that is, they generate the same lattice as the rows of the matrix*

$$M' = \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}.$$

Here $h = f^{-1} * g \pmod q$ in $\mathbb{Z}[X]/(X^N - 1)$ as usual.

Proof. It suffices to show that the transformation matrix $H = M'M_q^{-1}$ has determinant 1 and that H has entries in $\mathbb{Z}[X]/(X^N-1)$. We know $\det(M_q) = \det(M') = q$, so $\det(H) = 1$. Next we compute

$$\begin{aligned} H = \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix} \begin{pmatrix} f & g \\ F_q & G_q \end{pmatrix}^{-1} &= \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix} \begin{pmatrix} f & g \\ qF_1 & qG_1 \end{pmatrix}^{-1} \\ &= \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix} \begin{pmatrix} G_1 & -g/q \\ -F_1 & f/q \end{pmatrix} \\ &= \begin{pmatrix} G_1 - F_1 * h & (-g + f * h)/q \\ -qF_1 & f \end{pmatrix} \end{aligned}$$

It is clear that this last matrix has entries in $\mathbb{Z}[X]/(X^N-1)$, since $f*h \equiv g \pmod q$ by construction. Thus H is a unimodular transformation matrix, so the rows of M_q and M' generate the same lattice. □

The rows of the matrix M_q generate the NTRU lattice L_h, but they are not useful for directly finding close vectors, because the resultant construction leads to polynomials F_q and G_q whose coefficients are very large. We will make (F_q, G_q) smaller by reducing the centered norm (2).

The key to reducing the norm of the second row of M_q is to observe that we may alter it by any multiple of the first row, since this doesn't change the lattice generated by the rows. The polynomials f and g are already very small, and their centered rotations are reasonably orthogonal to one another, and so this reduction works quite well. (One may view this process as solving appr-CVP with a partial good basis.)

To explain how this reduction works and measure its effectiveness, we reuse some information from the earlier resultant computation (26,27). We begin with the first column of M_q. We would like to subtract a multiple $k * f$ of f from F_q to make it small. The "inverse technique" for solving appr-CVP says to take $k \approx F_q * f^{-1}$,

where f^{-1} is the inverse of f in $\mathbb{Q}[X]/(X^N-1)$. Using the resultant relation (26), we see that

$$f^{-1} = \frac{u}{R_f} \in \mathbb{Q}[X]/(X^N-1),$$

so we set $k = \lfloor u/R_f \rceil$. That is, $k \in \mathbb{Z}[X]/(X^N-1)$ is obtained by rounding the coefficients of u/R_f to the nearest integer. With this choice of k, we set $F = F_q - k*f$. Then F satisfies

$$F = F_q - k*f = F_q - \lfloor F_q * f^{-1} \rceil * f = \big(F_q * f^{-1} - \lfloor F_q * f^{-1} \rceil\big) * f = \epsilon_f * f,$$

where ϵ_f has coefficients in the interval $[-1/2, 1/2]$. Further, the coefficients of ϵ_f will be more-or-less uniformly and independently distributed within this interval, so on average we have $\|\epsilon_f\| \approx \sqrt{N/12}$. Hence

$$\|F\| \approx \|\epsilon_f\| \cdot \|f\| \approx \|f\| \sqrt{\frac{N}{12}}.$$

This shows that we can make the lower lefthand entry of M_q small, but we want to simultaneously do the same thing for the lower righthand entry. In other words, we want to subtract a multiple of g from G_q to get a small result. The problem is that if we replace F_q with $F_q - k*f$, then we are required to replace G_q with $G_q - k*g$, and we must use the same k. On the other hand, we know that the "right" k to use for G_q is $k \approx G_q * g^{-1}$. It turns out that the right k for F_q and the right k for G_q are almost equal to one another. To see why this is true, we divide the relation (29) by $f*g$, which yields

$$\frac{G_q}{g} = \frac{F_q}{f} + \frac{q}{f*g}.$$

Hence the difference between $G_q * g^{-1}$ and $F_q * f^{-1}$ has size approximately

$$\left\| \frac{G_q}{g} - \frac{F_q}{f} \right\| = \left\| \frac{q}{f*g} \right\| \approx \frac{q}{\|f\| \cdot \|g\|}.$$

For practical sets of parameters (12), this quantity is equal to 2.49, which means that on average, the coefficients of $G_q * g^{-1}$ and $F_q * f^{-1}$ differ by 0.157. Combining the results of this section, we have proven the following result:

Proposition 3. *Let $f, g \in \mathbb{Z}[X]/(X^N-1)$ and $h = f^{-1} * g \pmod q$ be as usual. Let $F_q, G_q \in \mathbb{Z}[X]/(X^N-1)$ satisfy*

$$f * G_q - g * F_q = q.$$

Set

$$k = \left\lfloor \frac{F_q * f^{-1} + G_q * g^{-1}}{2} \right\rceil, \quad F = F_q - k*f, \quad G = G_q - k*g. \tag{30}$$

Then the rows of the matrix

$$M = \begin{pmatrix} f & g \\ F & G \end{pmatrix}$$

generate the NTRU lattice L_h, and F and G have norms

$$\|F\| \approx \|f\| \sqrt{\frac{N}{12}} \qquad \|G\| \approx \|g\| \sqrt{\frac{N}{12}}.$$

Remark 15. The size of the private basis determines how well we can solve the approximate CVP. For this reason, it is useful in practice to further subtract multiples of (f, g) from (F, G) to try to reduce the norm. We thus perform a last stage of reduction by testing, for $i = 0, 1, 2, ..., B$, whether the vector

$$(F, G) - X^i * (f, g)$$

is smaller than (F, G). If it is, we replace (F, G) by this smaller vector and continue. Note that we may find improvements even with $B > N$, since the modifications made by the subsequent values of i may mean that a previously used value of i is again able to reduce the norm.

Remark 16. The resultant calculation yields a large vector (F, G). We have made this vector much smaller by modifying it using the formulas given in (30). This corresponds to treating F and G separately. In practice, one obtains a smaller result by treating them together, although this does require some additional resultant calculations. We omit the derivation and simply note that the desired value for k is

$$k = \left\lfloor \frac{\bar{f} \cdot F + \bar{g} \cdot G}{\bar{f} \cdot f + \bar{g} \cdot g} \right\rceil,$$

where $\bar{f}$ and $\bar{g}$ denote the reversals of f and g, that is $\bar{f}(X) = f(X^{-1})$ and $\bar{g}(X) = g(X^{-1})$.

6. Optimizations and Enhancements

In this section we discuss various methods that can be used to make NTRUSIGN even more efficient. Some of these methods have security implications, so they must be anaylzed with care before being used in any particular application.

6.1. **Reduced message digest.** Rather than use message digests of the form (m_1, m_2) it turns out to be more efficient to use ones of the form $(0, m)$. In this section we explain the basic properties of the NTRU lattice that mean that such a change has no detrimental impact on security; indeed it may be preferred.

Thus we ask whether a transcript of signatures on message digests of the form $(0, m)$ might leak more information that signatures on arbitrary points (m_1, m_2). The key observation is to note that if (s, t) is an NTRUSIGN signature on a point (m_1, m_2), then for any lattice point (x, y), the lattice point $(s+x, t+y)$ is a signature on the point $(m_1 + x, m_2 + y)$. Taking

$$(x, y) = -(m_1, m_1 * h \bmod q),$$

we see that any signature can be transformed into a signature on a message digest of the form $(0, m)$. Thus signing message digests of the form $(0, m)$ provides no more information to the attacker than does signing general message digests of the form (m_1, m_2).

We also observe that although the space of pairs (m_1, m_2) has size q^{2N}, the corresponding set of pairs (a, A) that determine the signature only has size q^N, as explained in Remark 14. Using message digests of the form $(0, m)$ means that the message digests are in one-to-one correspondence with the signatures that are created using a given basis (f, g, F, G).

6.2. **Product forms for message digests and keys.** For the original NTRU public key cryptosystem, significant efficiency gains can be achieved by taking some polynomials (e.g., m, f, g) to be products of several sparse binary polynomials, see [13]. Such an approach clearly reduces the search space, but if implemented properly, it appears to give an attacker no other advanatge. Of course the search space must be designed to be large enough to defeat meet-in-the-middle attacks.

In a similar way, it is possible to increase the efficiency of NTRUSIGNby taking certain polynomials to be products. For example, one might take f, g and/or m to be products of sparse binary polynomials.

6.3. **Computation of centered norms with mod q reduction.** When computing the centered norm of a "small" mod q vector, it is important to center the vector properly. Thus for example, the verifier only knows t modulo q, and if the center of the true t (i.e., the correct value of t not reduced mod q) is near to $q/2$ or $-q/2$, then the centered norm will not be correct unless the coefficients of t are shifted before being reduced modulo q. There are two ways to do this. An inefficient way is to look at all of the shifts of t, reduce each one modulo q, and check which one gives the smallest norm. A better approach is to find the largest string of consecutive values mod q that do not appear as coefficients of t, and shift t so that the middle of this string becomes $-q/2$. A third approach would be to use the fact that one knows the values of various polyomials at $X = 1$ to deduce where the centering value should lie. This third approach is the one used in NTRUENCRYPT during the decryption process.

The following is a practical definition of $\|t\|$, where we assume q is even.

- Compute $\kappa = N^{-1} * t(1) + \frac{q}{2} \pmod q$ with $0 \le \kappa < q$, where N^{-1} is the inverse of N modulo q.
- Create a new polynomial $\bar{t}$ with integer coefficients by the rule

$$\bar{t}_i = \begin{cases} t_i & \text{if } t_i < \kappa, \\ t_i + q & \text{if } t_i \ge \kappa. \end{cases}$$

- Define the *centered mod q norm of t* to be $\|t\| = \|\bar{t}\|$.

6.4. **Ensuring Valid Signatures.** Because signatures lie in a parallelopiped whose longest diagonal is greater than `NormBound`, it is theoretically possible for a validly generated signature to fail the verification test. The chance of this happening appears extremely small: from experiments, extrapolating results from a set of 1,000,000 signatures and using `NormBound` $= 300$, the chance of failure ranges from 10^{-12} with $|F, G| = 45$ to 10^{-18} with $|F, G| = 40$.

For greatest efficiency, a signer may choose to accept this small probability of verification failure. Alternatively, the signer may use the following or a similar algorithm to generate signatures:

(1) Initialize the counter c to 0.
(2) Compute the hash h of the document using function H_1.
(3) Generate (m_1, m_2) using function H_2 with input $(h||c)$, "||" denoting concatenation.
(4) Generate the signature (s, t) on (m_1, m_2).
(5) Attempt to verify the signature (s, t). If verification succeeds, output s as the signature. If it fails, increment c by 1 and go to step 3.

Even if c is only a single byte long, this modified signature generation makes the chance of failure negligibly small. The price paid is an additional verification operation when signing. We anticipate that implementors will decide themselves whether or not to include this additional check depending on application-specific requirements.

6.5. **Using the transpose lattice.** If one uses the product optimization in Section 6.2, then a further optimization may be possible by a simple modification of the key generation algorithm described in Section 5. The idea is to use the transpose of the matrix of determinant q. The advantage is that we can choose f and g to have product structures, but we cannot force F_q and G_q to be products. Note that we still end up with a public key h and the usual NTRU lattice L_h in Hermite Normal Form.

In the earlier construction, we ended up with a basis consisting of one very short vector (f, g) and one moderately short vector (F, G). In this new transpose construction, we will obtain a basis (f, F) and (g, G) in which the two basis vectors are about the same size, but their first coordinates are short and their second coordinates are larger. Given such a basis, it would be natural to weight the norm so as equalize the contributions of the two halves.

More work needs to be done examining the security of this new lattice, but there are clear efficiency gains from using it. There may also be advantages for security, since appr-CVP can be solved better with a "square" basis than with a "rectangular" basis. However, we note that in the transpose lattice the quantities on the left hand side are smaller than in the NTRU lattice case, and so averages over these quantities will tend to converge faster. It might be possible to overcome this problem by perturbing the message to be signed, as described in Section 6.6; however, this requires further analysis.

6.6. **Perturbations of m.** We have seen that although a transcript of 10 to 20 thousand signatures will reveal some lattice information via the use of second moments, the most effective known attack exploiting that information is at least as hard as the original NTRU Lattice Key Problem. We now describe a simple modification in the signing technique that, at a small efficiency cost, appears to eliminate both the linearity and the matrix product structure from any second moment transcript analysis.

The method involves perturbing m and depends on our earlier observation that a signature (s, t) on m will also sign all nearby points. The signer begins by choosing a short (secret) vector $w = (w_1, w_2)$ and storing it as part of his private key. Then the signer modifies the signing process as follows:

(1) Compute the vector $m = (m_1, m_2)$ as usual.
(2) Choose a random short vector $r = (r_1, r_2)$ satisfying $r_1(1) = r_2(1) = 0$.
(3) Let $z = (z_1, z_2) = (r_1 * w_1, r_2 * w_2)$ and compute the signature (s, t) for the vector $m + z$.

If r and w are short, say so that $\|z\| = O(\sqrt{N})$, then it is easy to check that the signature (s, t) on $m + z$ will also be a signature for m.

Now consider what information is revealed by averaging the second moments of a transcript. The transcript consists of sample vectors of the form

$$\epsilon_f * f + \epsilon_F * F + r_1 * w_1 \quad \text{and} \quad \epsilon_g * g + \epsilon_G * G + r_2 * w_2.$$

The first moments yield no information (since $r_1(1) = r_2(1) = 0$ and the rs are randomly chosen). The second moments yield quantities of the form

$$c_1 * (f * \bar{f} + F * \bar{F}) + c_2 * w_1 * \bar{w}_1 \quad \text{and} \quad c_1' * (g * \bar{g} + G * \bar{G}) + c_2' * w_2 * \bar{w}_2$$

analogous to those described in Section 4.6, where c_1 and c_2 are known constants. The appearance of the extra unknown quantities w_1 and w_2 means that the attacker does not recover a Gram matrix. Thus even if it is possible to effectively solve the Gram Factorization Problem, this method precludes its use in a second moment attack.

Of course, if the attacker can compute fourth moments sufficiently accurately, then he can eliminate w_1 and w_2 from the above equations. But we have seen that fourth moment averages converge extremely slowly. And in order to directly extract (say) $f * \bar{f}$ from the transcript, the attacker would actually need to compute sixth moments.

Finally, we remark that the idea described in this section can be considerably generalized, or otherwise altered to improve efficiency. A simple generalization is to take several vectors $w, w', w'', \ldots$ and sign $m + r * w + r' * w' + r'' * w'' + \cdots$. In full generality, one could select z from a large collection of vector-valued random variables with the property that the moments of z conceal the lattice-information revealed by the average moments of a transcript of signatures. Ideally, one would like the signature vectors to be uniformly distributed on a sphere. It is not possible to do this exactly, and possibly not even arbitrarily closely (remember that the z vectors cannot be too long), but even a partial approximation to a sphere could greatly reduce the effectiveness of transcript analysis.

7. History of NSS

In this section we briefly review the history of an earlier digital signature scheme, called NSS, that was based on NTRU lattices. We do so to illustrate how a flawed method of attaching the digital document to the signature led to serious weaknesses in NSS, and to explain how the direct and straightforward linkage of NTRUSign signatures to the underlying appr-CVP differentiates NTRUSign from NSS and frees it from the weaknesses of the the earlier scheme.

The NSS encoding method, as described at Eurocrypt 2001 [9], embedded the digital document directly into the signature via the use of auxiliary congruence conditions modulo a small prime p. It was discovered indepedently by Gentry, Jonsson, and Stern (see [3]) that weakness in this encoding method could be used to directly forge signatures without knowledge of the private key. At the same time, Szydlo (also described in [3]) found another way to exploit congruence conditions that allowed him recover the key from a long transcript of valid signatures.

The source of these weaknesses was an incomplete linking of the NSS method with the (approximate) closest vector problem in the NTRU lattice. More precisely, a signature exhibited a solution to a closest vector problem in the following sense: a point in space was associated to a message digest and a point in the NTRU lattice was constructed, using a knowledge of the private key (a short vector in the lattice), that was reaonably close to the message digest point. To be accepted as a valid signature a lattice point had to be sufficiently close to the message digest point and its coordinates had to satisfy certain statistical properties modulo the auxiliary prime p.

In [9] the specified distance was small enough to rule out forgery by lattice reduction methods, but not small enough to rule out forgery by a different approach. This approach, of Gentry, Jonsson, and Stern, also succeeded in bypassing the mod p statistical condition. In addition, Szydlo noted that the specific method by which the signature point was created allowed secret key information to be extracted from a long transcript of signatures by analysis of certain frequency distributions.

At the presentation of [9], the authors described how simple modifications to the encoding technique eliminated the possibility of attacks from these directions. In particular, the relation between the signature point and the message digest point was tied far more closely to the closest vector problem, effectively eliminating any possibility of direct forgeries. However, a somewhat ad hoc method was still used to force the signature point to satisfy the necessary mod p statistical properties.

At the rump session of Crypto '01, Gentry and Szydlo presented a result (since published as [4]) demonstrating that a transcript of valid signatures produced by the modified NSS method still leaked enough information to effectively cut the key size in half. They also described a method that potentially exploited the leaked information to recover the full key in polynomial time. Their method exploited a weakness in the specific method by which an NSS signature vector was constructed from the message digest and the private key. A special form was imposed on the NSS private key f and public key h which allowed (approximate) CVP to be solved quickly and efficiently. However, this meant that the signature was closely related to a product $f * w$. Gentry and Szydlo succeeded in recovering $f * w$ and a related product $f * \tilde{f}$ from the signature, and knowledge of these two products gave them sufficient information to recover f in polynomial time.

The principle upon which NSS was based was very simple. The signer has private knowledge of short vectors in the NTRU lattice. Given an arbitrary point in space arising from a message digest, the signer uses this knowledge to find a point in the NTRU lattice close to the message point. He then exhibits this solution to the approximate CVP as his signature (cf. the GGH scheme [5]).

The weaknesses of NSS arose from the fact that the signer did not possess a complete basis of short vectors for the NTRU lattice L_h^{NT}. Instead he possessed knowledge of one short $2N$-dimensional vector (f, g) whose rotations spanned half the lattice. Because his knowledge was only partial, the signer created a rather weak solution to the approximate CVP by a specific construction. The non-general nature of this construction opened the door to the attacks described earlier.

In direct contrast to NSS, the link in NTRUSIGN between the signature and the underlying appr-CVP is clear and direct. The private (f, g) vector of a general NTRU lattice is first used to construct a complete short basis for the lattice. A message digest is then mapped to an arbitrary point in $2N$-dimensional space. The signature is a lattice point that is very close to the message digest point and is found in a generic way using the full basis of short vectors for the NTRU lattice. Thus the signature provides a direct and very sharp answer to the general (approximate) closest vector problem in the NTRU lattice. There are no additional tests to perform, no auxiliary mod p conditions to verify. Verification consists simply of checking that the signature point is in the NTRU lattice and that it is within the required distance of the message digest point.

References

[1] E.F. Brickell and K.S. McCurley. *Interactive Identification and Digital Signatures*, AT&T Technical Journal, November/December, 1991, 73–86.

[2] C. Gentry,*Key recovery and meesage attacks on NTRU-composite*, Advances in Cryptology—Eurocrypt '01, Lecture Notes in Computer Science, Springer-Verlag, 2001.

[3] Craig Gentry, Jakob Jonsson, Jacques Stern, Michael Szydlo *Cryptanalysis of the NTRU Signature Scheme (NSS) from Eurocrypt 2001*, Advances in Cryptology—Asiacrypt '01, Lecture Notes in Computer Science, Springer-Verlag, 2001.

[4] C. Gentry,M Szydlo*Cryptanalysis of the Revised NTRU Signature Scheme*, Advances in Cryptology—Eurocrypt '02, Lecture Notes in Computer Science, Springer-Verlag, 2002.

[5] O. Goldreich, S. Goldwasser, S. Halevi, *Public-key cryptography from lattice reduction problems*. In Proc. CRYPTO'97, Lect. Notes in Computer Science 1294, Springer-Verlag, 1997, 112–131.

[6] L.C. Guillou and J.-J. Quisquater. *A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory*, Advances in Cryptology—Eurocrypt '88, Lecture Notes in Computer Science 330 (C.G. Günther, ed.), Springer-Verlag, 1988, 123–128.

[7] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. Silverman, W. Whyte *Enhanced Encoding and Verification Methods for the NTRU Signature Scheme*, NTRU Technical Note #18, *** 2002, `<www.ntru.com>`.

[8] J. Hoffstein, J. Pipher, J.H. Silverman, *NTRU: A new high speed public key cryptosystem*, in Algorithmic Number Theory (ANTS III), Portland, OR, June 1998, Lecture Notes in Computer Science 1423 (J.P. Buhler, ed.), Springer-Verlag, Berlin, 1998, 267–288.

[9] J. Hoffstein, J. Pipher, J.H. Silverman, *NSS: An NTRU Lattice-Based Signature Scheme*, Advances in Cryptology—Eurocrypt '01, Lecture Notes in Computer Science, Springer-Verlag, 2001.

[10] J. Hoffstein, J. Pipher, J.H. Silverman, *Enhanced Encoding and Verification Methods for the NTRU Signature Scheme*, NTRU Technical Note #017, May 2001, `<www.ntru.com>`.

[11] J. Hoffstein, D. Lieman, J.H. Silverman, *Polynomial Rings and Efficient Public Key Authentication*, in Proceeding of the International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC '99), Hong Kong, (M. Blum and C.H. Lee, eds.), City University of Hong Kong Press.

[12] J. Hoffstein, J.H. Silverman, *Polynomial Rings and Efficient Public Key Authentication II*, in Proceedings of a Conference on Cryptography and Number Theory (CCNT '99), (I. Shparlinski, ed.), Birkhauser.

[13] J. Hoffstein, J.H. Silverman, *Optimizations for NTRU*, Public-Key Cryptography and Computational Number Theory (Warsaw, September 11-15, 2000), DeGruyter, to appear.

[14] H. Koy, C.-P. Schnorr, *Segment LLL-Reduction of Lattice Bases*, Cryptography and Lattices Conference–Proceedings of CaLC '01, (March 2001, Providence, RI), J. Silverman (ed.), Lecture Notes in Computer Science, Springer-Verlag, 67–80.

[15] H. Koy, C.-P. Schnorr, *Segment LLL-Reduction with Floating Point Orthogonalization*, Cryptography and Lattices Conference–Proceedings of CaLC '01, (March 2001, Providence, RI), J. Silverman (ed.), Lecture Notes in Computer Science, Springer-Verlag, 81–96.

[16] J. Lagarias, H.W. Lenstra, Jr., C.P. Schnorr, *Korkin-Zolotarev bases and successive minima of a lattice and its reciprocal lattice*, Combinatorica 10 (1990), 333–348

[17] A.K. Lenstra, H.W. Lenstra Jr., L. Lovász, *Factoring polynomials with rational coefficients*, Mathematische Ann. 261 (1982), 513–534.

[18] A. May, J.H. Silverman, *Dimension reduction methods for convolution modular lattices*, Cryptography and Lattices Conference–Proceedings of CaLC '01, (March 2001, Providence, RI), J. Silverman (ed.), Lecture Notes in Computer Science, Springer-Verlag, 110–125.

[19] A.J. Menezes and P.C. van Oorschot and S.A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1996.

[20] D. Micciancio, *Improving lattice based cryptosystems using Hermite normal form*, Cryptography and Lattices Conference–Proceedings of CaLC '01, (March 2001, Providence, RI), J. Silverman (ed.), Lecture Notes in Computer Science, Springer-Verlag, 126–145.

[21] P. Nguyen, *Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto '97*, Advances in Cryptology—Proceedings of CRYPTO '99, (August 15–19, 1999, Santa Barbara, California), M. Wiener (ed.), Lecture Notes in Computer Science, Springer-Verlag.

[22] T. Okamoto. *Provably secure and practical identification schemes and corresponding signature schemes*, Advances in Cryptology—Crypto '92, Lecture Notes in Computer Science 740 (E.F. Brickell, ed.) Springer-Verlag, 1993, 31–53.

[23] A. Renvall, University of Turku, private communication.

[24] C.-P. Schnorr, *A hierarchy of polynomial time lattice basis reduction algorithms*, Theoretical Computer Science 53 (1987), 201–224.

[25] C.-P. Schnorr, *A more efficient algorithm for lattice basis reduction*, J. Algorithms 9 (1988), 47–62.

[26] C.-P. Schnorr. *Efficient identification and signatures for smart cards*, Advances in Cryptology—Crypto '89, Lecture Notes in Computer Science 435 (G. Brassard, ed), Springer-Verlag, 1990, 239–251.

[27] C.-P. Schnorr, M. Euchner, *Lattice basis reduction: improved practical algorithms and solving subset sum problems*, Math. Programming 66 (1994), no. 2, Ser. A, 181–199.

[28] J. Stern. *A new identification scheme based on syndrome decoding*, Advances in Cryptology—Crypto '93, Lecture Notes in Computer Science 773 (D. Stinson, ed.), Springer-Verlag, 1994, 13–21.

[29] J. Stern. *Designing identification schemes with keys of short size*, Advances in Cryptology—Crypto '94, Lecture Notes in Computer Science 839 (Y.G. Desmedt, ed), Springer-Verlag,1994, 164–173.

[30] D. Stinson, *Cryptography: Theory and Practice*. CRC Press, 1997.

NTRU CRYPTOSYSTEMS, INC., 5 BURLINGTON WOODS, BURLINGTON, MA 01803 USA

E-mail address: jhoffstein@ntru.com, NHowgraveGraham@ntru.com, jpipher@ntru.com, jsilverman@ntru.com, WWhyte@ntru.com