

NSS: The NTRU Signature Scheme

Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman

NTRU Cryptosystems, Inc., 5 Burlington Woods, Burlington, MA 01803 USA,
jhoff@ntru.com, jpipher@ntru.com, jhs@ntru.com

Abstract. A new authentication and digital signature scheme called the NTRU Signature Scheme (NSS) is introduced. NSS provides an authentication/signature method complementary to the NTRU public key cryptosystem. The hard lattice problem underlying NSS is similar to the hard problem underlying NTRU, and NSS similarly features high speed, low footprint, and easy key creation.

Keywords: digital signature, public key authentication, lattice

Introduction

Secure public key authentication and digital signatures are increasingly important for electronic communications and commerce, and they are required not only on high powered desktop computers, but also on Smart-Cards and wireless devices with severely constrained memory and processing capabilities. 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, 2, 5, 6, 8–10, 13–15].

At CRYPTO '96 the authors introduced a highly efficient new public key cryptosystem called NTRU. (See [4] for details.) Underlying NTRU is a hard mathematical problem of finding short vectors in certain lattices. In this note we introduce a complementary fast authentication and digital signature scheme that uses public and private keys of the same form as those used by the NTRU public key cryptosystem. We call this new algorithm NSS for NTRU Signature Scheme.

The authors would like to thank Phil Hirschorn for much computational assistance and Don Coppersmith for substantial help in analyzing the security of NSS. Any remaining weaknesses or errors in the signature scheme described below are, of course, entirely the responsibility of the authors.

1 A Brief Description of NSS

In this section we briefly describe NSS, the NTRU Signature Scheme. In order to avoid excessive duplication of exposition, we will assume some familiarity with [4], but we will repeat definitions and concepts when it appears that this would be useful. Thus this paper should be readable without reference to [4].

The basic operations occur in the ring of polynomials

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

of degree $N - 1$, where multiplication is performed using the rule $X^N = 1$. The coefficients of these polynomials are then reduced modulo p or modulo q , where p and q are fixed integers.

There are five integer parameters associated to NSS,

$$(N, p, q, D_{\min}, D_{\max}).$$

There are also several sets of polynomials $\mathcal{F}_f, \mathcal{F}_g, \mathcal{F}_w, \mathcal{F}_m$ having small coefficients that serve as sample spaces. For concreteness, we mention the choice of integer parameters

$$(N, p, q, D_{\min}, D_{\max}) = (251, 3, 128, 55, 87), \quad (1)$$

which appears to yield a secure and practical signature scheme. See Section 2 for further details.

Remark 1. For ease of exposition we will often assume that $p = 3$. We further assume that polynomials with mod q coefficients are chosen with coefficients in the range $-q/2$ to $q/2$.

The public and private keys for the NTRU Signature Scheme (NSS) are formed as follows. Bob begins by choosing two polynomials f and g having the form

$$f = f_0 + pf_1 \quad \text{and} \quad g = g_0 + pg_1. \quad (2)$$

Here f_0 and g_0 are fixed universal polynomials (e.g., $f_0 = 1$ and $g_0 = 1 - X$) and f_1 and g_1 are polynomials with small coefficients chosen from the sets \mathcal{F}_f and \mathcal{F}_g , respectively. Bob next computes the inverse f^{-1} of f modulo q , that is, f^{-1} satisfies

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

Bob's public verification key is the polynomial

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

Bob's private signing key is the pair (f, g) .

Before describing exactly how NSS works, we want to explain the underlying idea. The coefficients of the polynomial h have the appearance of being random numbers modulo q , but Bob knows a small polynomial f with the property that the product $g \equiv f * h \pmod{q}$ also has small coefficients. Equivalently (see Section 4.3), Bob knows a short vector in the NTRU lattice generated by h . It is a difficult mathematical problem, starting from h , to find f or to find some other small polynomial F with the property that $G \equiv F * h \pmod{q}$ is small. Bob's signature s on a digital document D will be linked to D and will demonstrate to Alice that he knows a decomposition $h \equiv f^{-1} * g \pmod{q}$ without giving Alice information that helps her to find f . The mechanism by which Bob shows that he knows (f, g) without actually revealing their values lies at the heart of NSS and is described in the next section.

1.1 NSS Key Generation, Signing, and Verifying

We now describe in more detail the steps used by Bob to sign a document and by Alice to verify Bob's signature. The key computation involves the *deviation* between two polynomials. Let $a(X)$ and $b(X)$ be two polynomials in R . We first reduce their coefficients modulo q to lie in the range between $-q/2$ to $q/2$, then we reduce their coefficients modulo p to lie in the range between $-p/2$ and $p/2$. Let

$$A(X) = A_0 + A_1X + \cdots + A_{N-1}X^{N-1} \text{ and } B(X) = B_0 + \cdots + B_{N-1}X^{N-1}$$

be these reduced polynomials. Then the *deviation of a and b* is

$$\text{Dev}(a, b) = \#\{i : A_i \neq B_i\}.$$

Intuitively, $\text{Dev}(a, b)$ is the number of coefficients of $a \pmod{q}$ and $b \pmod{q}$ that differ modulo p .

Key Generation: This was described above, but we briefly repeat it for convenience. Bob chooses two polynomials f and g having the appropriate form (2). He computes the inverse f^{-1} of f modulo q . Bob's public verification key is the polynomial $h \equiv f^{-1} * g \pmod{q}$ and his private signing key is the pair (f, g) .

Signing: Bob’s document is a polynomial m modulo p . (In practice, m must be the hash of a document, see Section 4.10.) Bob chooses a polynomial $w \in \mathcal{F}_w$ of the form

$$w = m + w_1 + pw_2,$$

where w_1 and w_2 are small polynomials whose precise form we will describe later. He then computes

$$s \equiv f * w \pmod{q}.$$

Bob’s signed message is the pair (m, s) .

Verification: In order to verify Bob’s signature s on the message m , Alice first checks that $s \neq 0$ and then she verifies the following two conditions:

(A) Alice compares s to $f_0 * m$ by checking if their deviation satisfies

$$D_{\min} \leq \text{Dev}(s, f_0 * m) \leq D_{\max}.$$

(B) Alice uses Bob’s public verification key h to compute the polynomial

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

putting the coefficients of t into the range $[-q/2, q/2]$ as usual. She then checks if the deviation of t from $g_0 * m$ satisfies

$$D_{\min} \leq \text{Dev}(t, g_0 * m) \leq D_{\max}.$$

If Bob’s signature passes tests (A) and (B), then Alice accepts it as valid.

We defer until Section 3 below a detailed explanation of why NSS works. However, we want to mention here the reason for allowing s and t to deviate from $f_0 * m$ and $g_0 * m$, respectively. This permits us to take w_1 to be nonzero and to allow a significant amount of reduction modulo q to occur in the products $f * w$ and $g * w$. This makes it difficult for an attacker to find the exact values of $f * w$ or $g * w$ over \mathbb{Z} , which in turn means that potential attacks via lattice reduction require lattices of dimension $2N$ rather than N . However, it is certainly also possible to set up a version of NSS without deviations and to allow a transcript to reveal $f * w$ and $g * w$ exactly. If N is chosen greater than 680 this still gives a fast and equally secure signature scheme, albeit with somewhat larger key and signature sizes than the version of NSS described in this note.

The check by Alice that $s \neq 0$ is done to eliminate the small possibility of a forgery via the trivial signature. This is described in more detail in Appendix C.

This concludes our overview of how NSS works. In the remaining sections we will analyze NSS more closely, provide a security analysis, and suggest a parameter set providing a suitable level of security. For a comparison of NSS with the NTRU public key cryptosystem, see Appendix F.

2 A Practical Implementation of NSS

Based on the security analysis in Section 4 below, the following parameter selection for NSS appears to provide a security level at least as high as an RSA 1024 bit modulus:

$$(N, p, q, D_{\min}, D_{\max}) = (251, 3, 128, 55, 87). \quad (3)$$

This leads to the following key and signature sizes for NSS:

Public Key: 1757 bits Private Key: 502 bits Signature: 1757 bits

We take $f_0 = 1$ and $g_0 = 1 - 2X$, where recall that $f = f_0 + pf_1$ and $g = g_0 + pg_1$. In order to describe the sample spaces, we let

$$\mathcal{T}(d) = \{F(X) \in R : F \text{ has } d \text{ coefs} = 1 \text{ and } d = -1, \text{ with the rest } 0\}.$$

Then the sample spaces corresponding to the parameter set (3) are

$$\mathcal{F}_f = \mathcal{T}(70), \quad \mathcal{F}_g = \mathcal{T}(40), \quad \mathcal{F}_m = \mathcal{T}(32).$$

Note that m is a hash of the digital document D being signed. Thus the users must agree on a method (e.g., using SHA1) to transform D into a list of 64 distinct integers $0 \leq e_i < 251$, and then $m = \sum_{i=1}^{32} X^{e_i} - \sum_{i=33}^{64} X^{e_i}$.

We also must explain how to choose the polynomials w_1 and w_2 . This must be done carefully so as to prevent an attacker from either lifting to a lattice over \mathbb{Z} (see Section 4.5) or gaining information via a reversal averaging attack (see Section 4.7). Roughly, the idea is to choose random w_2 , compute $s' \equiv f * (m + pw_2) \pmod{q}$ and $t' \equiv g * (m + pw_2) \pmod{q}$, choose w_1 to cancel all of the common deviations of $(s', f_0 * m)$ and $(t', g_0 * m)$ and to exchange some of the noncommon deviations, and finally to alter w_2 to move approximately $1/p$ of the nonzero coefficients of $m + w_1$. The precise prescription is described using C pseudo-code in Appendix G, see Figure 1.

Table 2 in Appendix H gives the results of experiments showing that Bob’s signature will generally (but not always) be valid, so Bob should definitely check the signature before publishing it and in those cases that it fails test (A) or (B), he should choose a different w_1 and w_2 and try again. Note that the probability estimates given in Section 4.2 (see also Table 3 in Appendix H show that there is only a tiny chance, smaller than 2^{-80} , that a randomly chosen s' satisfying test (A) will also satisfy test (B).

We have implemented NSS in C and run it on various platforms. Table 1 describes the performance of NSS on a desktop machine and on a constrained device and gives comparable figures for RSA and ECDSA signatures.

	Pentium	Palm
NSS Sign	0.35 ms	0.33 sec
RSA Sign	66.56 ms	36.13 sec
ECDSA Sign	1.18 ms	1.79 sec
NSS Verify	0.29 ms	0.25 sec
RSA Verify	1.23 ms	0.73 sec
ECDSA Verify	1.70 ms	3.26 sec

Table 1. Speed Comparison of NSS, RSA, and ECDSA

Notes for Table 1.

1. NSS speeds from the NERI implementation of NSS by NTRU Cryptosystems.
2. RSA and ECDSA speeds presented by Alfred Menezes [7] at CHES 2000.
3. RSA 1024 bit verify uses a small verification exponent for increased speed.
4. ECDSA 163 bit uses a Koblitz curve for increased speed. Time is approximately doubled if a random curve over $\mathbb{F}_{2^{163}}$ is used.

3 Completeness of NSS

A signature scheme is deemed to be complete if Bob’s signature, created with the private signing key f , will be accepted as valid. Thus we need to check that Bob’s signed message (m, s) passes the two tests (A) and (B).

Test (A): The polynomial s that Alice tests is congruent to the product

$$\begin{aligned} s &\equiv f * w \pmod{q} \\ &\equiv (f_0 + pf_1)(m + w_1 + p * w_2) \pmod{q} \\ &\equiv f_0 * m + f_0 * w_1 + pw_2 + pf_1 * w \pmod{q}. \end{aligned}$$

We see that i^{th} coefficients of s and $f_0 * m$ will agree modulo p unless one of the following situations occurs:

- The i^{th} coefficient of $f_0 * w_1$ is nonzero.
- The i^{th} coefficient of $f * w$ is outside the range $(-q/2, q/2]$, so differs from the i^{th} coefficient of s by some multiple of q .

If the parameters and sample spaces are chosen properly (e.g., as in Section 2) then there will be at least D_{\min} and at most D_{\max} deviations between $s \bmod p$ and $m \bmod p$. Thus Bob's signature passes test (A).

Test (B): The polynomial t is given by

$$t \equiv h * s \equiv (f^{-1} * g) * (f * w) \equiv g * w \pmod{q}.$$

Since g has the same form as f , the same reasoning as for test (A) shows that t will pass test (B).

Remark 2. We have indicated why, for appropriate choices of parameters, Bob's signature will probably be accepted by Alice. Note that when Bob creates his signature, he should check to make sure that it is a valid signature. Table 2 in Appendix H shows that for the parameters $(N, p, q, D_{\min}, D_{\max}) = (251, 3, 128, 55, 87)$ from Section 2, the probability that $\text{Dev}(s, f_0 * m)$ is valid is approximately 87.33% and the probability that $\text{Dev}(t, g_0 * m)$ is valid is approximately 90.92%, so Bob's signature will be valid about 79.40% of the time. Of course, if it is not valid, he simply chooses a new random polynomial w_2 and tries again. In practice it will not take very many tries to find a valid signature. The timings given in Table 1 take this factor into account.

4 Security Analysis of NSS

It was shown in Section 3 that given a message m , Bob can produce a signature s satisfying the necessary requirements. In this section we will discuss various ways in which an observer Oscar might try to break the system. There are many attacks that he might try. For example, he might attempt to discover the private key f or a useful imitation, either directly from the public key h or from a long transcript of valid signatures. He

might also try to forge a signature on a message without first finding the private key. We will describe hard lattice problems that underlie some of these attacks and examine the probabilities of the other attacks that rely on random searches. In all cases we will explain why the indicated attacks are infeasible for an appropriate choice of parameters such as those given in Section 2. Due to space constraints, we will have to refer the reader to the appendices for many of the technical details.

4.1 The Norm of a Polynomial

In order to analyze the two verification conditions and relate them to both combinatorial problems and lattice problems, we briefly digress to discuss norms of polynomials. For simplicity, we will consider only polynomials in the subset

$$R^0 = \{a(X) \in R : a(1) = 0\}.$$

For more general polynomials, one should first center their coefficients around 0.

Let

$$a(X) = a_0 + a_1X + a_2X^2 + \cdots + a_{N-1}X^{N-1} \in R^0.$$

The *Euclidean Norm* (also called the L^2 norm) of a is denoted $\|a\|$ and defined by

$$\|a\|^2 = a_0^2 + a_1^2 + \cdots + a_{N-1}^2.$$

4.2 Random Search for a Valid Signature on a Given Message

Given a message m , Oscar must produce a signature s satisfying

- (A) $D_{\min} \leq \text{Dev}(s, f_0 * m) \leq D_{\max}$.
- (B) $D_{\min} \leq \text{Dev}(t, g_0 * m) \leq D_{\max}$, where $t \equiv s * h \pmod{q}$.

The most straightforward approach for Oscar is to choose s at random satisfying condition (A), which is obviously easy to do, and then to hope that t satisfies condition (B). If it does, then Oscar has successfully forged Bob's signature, and if not, then Oscar can try again with a different s . For a randomly selected s satisfying (A), the polynomial t will resemble a random polynomial modulo m , so we can compute the probability that t satisfies (B). For the parameters in Section 2, this probability turns out to be approximately 2^{-81} . For details, see Appendix A.

4.3 NTRU Lattices and Lattice Attacks on the NSS Public Key

Recall, (2), that the public key for our scheme has the form $h \equiv f^{-1} * g \pmod{q}$, where $f = f_0 + pf_1$ and $g = g_0 + pg_1$. As this is very similar to the form of an NTRU public key, a $2N$ dimensional lattice attack based on the shortest vector in as in [4] can be used to try to derive f, g from h . A more effective attack is to use the knowledge of f_0, g_0 to set up a closest vector attack on f_1, g_1 in a similar $2N$ dimensional lattice. This second attack is the most effective that we know of on the signature scheme. Extensive numerical experiments and idealized (for the attacker) extrapolations of current lattice reduction algorithms have established a conservative lower bound of 10^{12} MIPS years for a solution to this problem when $N = 251$. For further details, see Appendix A.1.

4.4 Lattice Attacks on Transcripts

Another potential area of vulnerability is a transcript of signed messages. Oscar can examine a list of signatures $s, s', s'' \dots$, which means that he has at his disposal the lists

$$fw, fw', fw'', \dots \pmod{q} \quad \text{and} \quad gw, gw', gw'', \dots \pmod{q}. \quad (4)$$

If Oscar can determine any of the w values, then he can easily recover f and g . Using division, Oscar can obtain $w^{-1}w' \pmod{q}$ and other similar ratios, so he can launch an attack on the pair (w, w') identical to that described in the preceding section. With this approach, the value of κ above is increased, making the breaking time considerably greater than 10^{12} MIPS-Years.

Oscar can also set up a kN dimensional NTRU type lattice using the ratios of signatures $w^{(1)}/w^{(1)}, w^{(2)}/w^{(1)}, \dots, w^{(k)}/w^{(1)}$. The target would then be $(w^{(1)}, w^{(2)}, \dots, w^{(k)})$. With this approach the κ value decreases as k increases, giving the attacker something of an advantage, but the increasing dimension more than offsets this. With the parameters given in Section 2, the optimum value of k for the attacker is $k = 10$, giving a ratio

$$\kappa = 4.87/\sqrt{10N}.$$

This is a bit better than the $c > 5.15$ coming from the original $2N$ dimensional lattice, but still considerably worse than the $c = 2.8936$ that gave us the original lower bound. Thus for any $k > 2$ we should end up

with a weaker lower bound for the breaking time than that given by the previously described search for F, G using the public key h .

There are several other variations on the lattice attacks described above, but the strongest appears to be the closest vector attack on the public key, described in the preceding section, with $\kappa > .23$.

4.5 Lifting a NSS Signature Lattice to \mathbb{Z}

An attacker Oscar is presumed to have access to a transcript of signed messages s, s', s'', \dots . This means that he can analyze lists of polynomials

$$f*w, f*w', f*w'', \dots \bmod q \quad \text{and} \quad g*w, g*w', g*w'', \dots \bmod q. \quad (5)$$

Various ways in which he might try to exploit this mod q information are described in Sections 4.4. In this section we will be concerned with the possibility that Oscar might lift the transcript information (5) and recover the values of $f*w, f*w', \dots$ exactly over \mathbb{Z} . The reason that this is of concern is that it would allow Oscar to create new lattices in which it is easier to find useful short vectors. For a description of these lattices, see Appendix B. The existence of these lattices shows that for the parameter set in Section 2, we must explain why it is not possible for Oscar to realistically lift the transcript (5) to \mathbb{Z} .

For each signature, Oscar has access to the three polynomials (s, t, m) . To exploit the transcript, Oscar studies the coefficients at which s deviates from m and t from $g_0 * m$. By comparing deviations, Oscar might hope to obtain information about the polynomial w_1 and about the coefficients at which $f*w$ and/or $g*w$ wrap modulo q . If w_1 were chosen at random, this approach might work, but as described in Section 2 (see also Figure 1 in Appendix G), the polynomial w_1 is not chosen at random. Instead, it is chosen to conceal and redistribute the deviations between s, t and m . This makes it infeasible for Oscar to gain enough useful information to lift s or t to \mathbb{Z} . For further details on this attack, see Appendix B.

We briefly mention two other approaches. Oscar might suspect that deviations near to $-q/2$ or $q/2$ are more likely to come from mod q wrapping than from w_1 . This is true to some extent, but in practice the wrapped coefficients stretch well past 0 in both directions. Finally, Oscar can use algebra to recover the polynomial $(g_0 * f_1 - f_0 * g_1) * w \bmod q$. If little reduction modulo q occurs, Oscar can lift to \mathbb{Z} and mount a strong lattice attack. In practice, there is a considerable amount of reduction in this polynomial. For further details on these last two approaches, see Appendix B.

4.6 Forgery Via Lattice Reduction

The opponent, Oscar, can try to forge a signature s on a given message m by means of lattice reduction. The best method that we have found for accomplishing this is described in Appendix C. It requires the successful analysis of a lattice of dimension $3N$ and appears to have a time requirement in excess of the 10^{12} MIPS years we have mentioned previously.

4.7 Transcript Averaging Attacks

As mentioned previously, examination of a transcript (4) of genuine signatures gives the attacker a sequence of polynomials of the form

$$s \equiv f * w \equiv (f_0 + pf_1)(m + w_1 + p * w_2) \pmod{q}$$

with varying w_1 and w_2 . A similar sequence is known for g . Because of the inherent linearity of these expressions, we must prevent Oscar from obtaining useful information via a clever averaging of long transcripts.

The primary tool for exploiting such averages is the *reversal* of a polynomial $a(X) \in R$ defined by $\rho(a) = a(X^{-1})$. Then the average of $a * \rho(a)$ over a sequence of polynomials with uncorrelated coefficients will approach the constant $\|a\|^2$, while the average of $a' * \rho(a)$ over uncorrelated polynomials will converge to 0. If m , w_1 , and w_2 were essentially uncorrelated, then Oscar could obtain useful information by averaging expressions like $s * \rho(m)$ over many signatures. Indeed, this particular expression would converge to $f\|m\|^2$, and thus would reveal the private key f .

There is an easy way to prevent all second moment attacks of this sort. Briefly, after m , w_1 , and a preliminary w_2 are chosen, Bob goes through the coefficients of $m + w_1$ and, with probability $1/p$, subtracts that value from the corresponding coefficient of w_2 . This causes averages of the form $a * \rho(b)$ created from signatures to equal 0. For further details on this attack and the defense that we have described, see Appendix D. We also mention that it might be possible to compute averages that yield the value of $f * \rho(f)$ and averages that use fourth power moments, but the former does not appear to be useful for breaking the scheme and the latter, experimentally, appears to converge much too slowly to be useful. Again we refer to Appendix D for details.

4.8 Forging Messages To Known Signatures

Another possible attack is to take a list of one or more valid signatures (s, t, m) , generate a large number of messages m' , and try to find a signature in the list that validly signs one of the messages. It is important to

rule out attacks of this sort, since for example, one might take a signature in which m says “IOU \$10” and try to find an m' that says “IOU \$1000”. Note that this attack is different from the attack in Section 4.2 in which one chooses an m and an s with valid $\text{Dev}(s, m)$ and hopes that $t \equiv h * s \pmod{q}$ has a valid $\text{Dev}(t, g_0 * m)$. The fact that (s, t, m) is already a valid signature implies some correlation between s and t , which may make it more likely that (s, t) also signs some other m' .

We ran experiments using the parameters from Section 2. The results displayed in Table 7 in Appendix H appear to show that the distribution of $\text{Dev}(s, m') + \text{Dev}(t, g_0 * m')$ is normally distributed with mean 283.54 and standard deviation 11.74. Using these values and extrapolating using a normal distribution, we find

$$\text{Prob}(\text{Dev}(s, m') + \text{Dev}(t, g_0 * m') \leq 174) \approx 2^{-67.4}.$$

This appears to provide adequate security for most situations, since note that even if the sum is smaller than $2 \cdot 87$, one is still not assured of having each piece smaller than 87.

For added security, Bob might reduce the value of D_{\max} to 81. This makes it only a little harder to produce a valid signature while reducing the above probability to approximately 2^{-82} .

4.9 Soundness of NSS

A signature scheme is considered sound if it can be proved that the ability to produce several valid signatures on random messages implies an ability to recreate the secret key. We can not prove this for the parameters given in Section 2, which have been chosen to maximize efficiency. Instead, the preceding sections on security analysis make a strong argument that forgery is not feasible without the private key, and that it is not feasible to recover the private key from either a transcript of valid signatures or the public key.

We can, however, make a probabilistic argument for soundness under certain assumptions. If we allow D_{\max} or q to be a bit smaller, and N to be a bit larger, then an argument applying the gaussian heuristic to a $4N$ dimensional lattice shows that a forged signature is highly probable to have the correct form, i.e., $s \equiv f * w \pmod{q}$. We can then demonstrate that the ability to generate a few such s for given messages m would allow one to recover f . For details, see Appendix E.

4.10 Signature Encoding

In practice, it is important that the signature be encoded (i.e., padded and transformed) so as to prevent a forger from combining valid signatures to produce new valid signatures. For example, let s_1 and s_2 be valid signatures on messages m_1 and m_2 , respectively. Then there is a small, but nontrivial, possibility that the sum $s_1 + s_2$ will serve as a valid signature for the message $m_1 + m_2$. This and other similar sorts of attacks are easily thwarted by encoding the signature. For example, one might start with the message M (which is itself probably the hash of a digital document) and concatenate it with a time/date stamp D and a random string R . Then apply an all-or-nothing transformation to $M\|D\|R$ to produce the message m to be signed using NSS. This allows the verifier to check that m has the correct form and prevents a forger from combining or altering valid signatures to produce a new valid signature.

This is related to the more general question of whether or not Oscar can create any valid signature pairs (m, s) , even if he does not care what the value of m is. When encoding is used, the probability that a random m will have a valid form can easily be made smaller than 2^{-80} .

Appendices

A Random Search for a Valid Signature on a Given Message

Given a message m , Oscar must produce a signature s satisfying

- (A) $D_{\min} \leq \text{Dev}(s, f_0 * m) \leq D_{\max}$.
- (B) $D_{\min} \leq \text{Dev}(t, g_0 * m) \leq D_{\max}$, where $t \equiv s * h \pmod{q}$.

The most straightforward approach for Oscar is to choose s at random satisfying condition (A), which is obviously easy to do, and then to hope that t satisfies condition (B). If it does, then Oscar has successfully forged Bob's signature, and if not, then Oscar can try again with a different s . Thus we must examine the probability that a randomly chosen s satisfying (A) will yield a t that satisfies (B).

The condition (A) on s has no real effect on the end result t , since t is formed by multiplying $s * h$ and reducing the coefficients modulo q , and the coefficients of h are essentially uniformly distributed modulo q . Thus we are really asking for the probability that a randomly chosen polynomial t with coefficients between $-q/2$ and $q/2$ will satisfy condition (B). This is easily computed using elementary probability theory.

The coefficients of a randomly chosen t can be viewed as N independent random variables taking values uniformly modulo q . The coefficients of m are fixed target values modulo p . We need to compute the probability that a randomly chosen N -tuple of integers modulo q has at least D_{\min} and no more than D_{\max} of its coordinates equal modulo p to fixed target values. Assuming that q is significantly larger than p , this probability is approximately

$$\text{Prob}(D_{\min} \leq \text{Dev}(t, g_0 * m) \leq D_{\max}) \approx \frac{1}{p^N} \sum_{d=D_{\min}}^{D_{\max}} \binom{N}{d} (p-1)^d.$$

Table 3 in Appendix H gives this probability for $(N, p) = (251, 3)$ and several values of D_{\min} and D_{\max} . For example, the table shows that for $D = 87$, the probability of a successful forgery using a randomly selected s is approximately $2^{-80.95}$.

A.1 NTRU Lattices and Lattice Attacks on the NSS Public Key

There are several possible lattice attacks on NSS. Oscar can try to extract the private key f from the public key h or from a long transcript of

genuine signatures. Alternatively, he can try to forge a signature without knowledge of f , using only h and a transcript. In this section we will discuss attempts by Oscar to obtain the private key from the public key by lattice reduction methods.

We have performed a large number of computer experiments to quantify the effectiveness of current lattice reduction techniques. This has given us a strong empirical foundation for analyzing and quantifying the vulnerability of several general classes of lattices to lattice reduction attacks. The following analysis and heuristics applies to the lattices discussed in this paper. (See also the lattice material in the papers [3–6].)

Let L be a lattice of determinant d and dimension n . Let v_0 denote a given fixed vector, possibly the origin. Let τ denote a given radius and consider the problem of locating a vector $v \in L$ such that $\|v - v_0\| < \tau$. The difficulty of solving this problem for large n is directly connected with the quantity

$$\kappa = \frac{\tau}{d^{1/n} \sqrt{n/(2\pi e)}}. \quad (6)$$

Here the denominator is the length that the gaussian heuristic predicts for the shortest expected vector in L . See [4] for a similar analysis.

If $\kappa < 1$ then by the gaussian heuristic a solution, if it exists, will probably be unique or almost unique. This solution will be easier to find by lattice reduction methods if κ is closer to 0, and harder to find if κ is closer to 1. In fact, if $L = L(n)$ and $v(n), v_0(n), L(n)$ is a sequence of lattices and target vectors with increasing n and

$$\kappa = \frac{c}{\sqrt{n}}. \quad (7)$$

for a constant c then it appears that the time necessary for lattice reduction methods to find $v(n)$ grows like $e^{\gamma n}$, with γ roughly proportional to c .

If $\kappa \geq 1$ then a solution will probably not be unique, but becomes progressively harder to find as κ approaches 1.

We must stress here that the above statements are not intended to be a proof of security, or to convey any assurance of security. They merely supply a conceptual framework that we have found useful for formulating working parameter sets. These parameter sets are then tested in detail.

The most straight forward attack on NSS is an attack on the key h using precisely the same $2N$ dimensional lattice used to attack NTRU keys. See [4, 11] for details on the NTRU lattice and the use of lattice reduction methods to compute the shortest expected vector. If we identify

polynomials with their vector of coefficients, then the $2N$ -dimensional NTRU lattice L^{NT} consists of the linear combinations of the $2N$ vectors in the set

$$\{(X^i, X^i * h) : 0 \leq i < N\} \cup \{(0, qX^i) : 0 \leq i < N\}.$$

Equivalently, L^{NT} is the set of all vectors $(F(X), F(X) * h(X))$, where $F(X)$ varies over all N -dimensional vectors and the last N coordinates are allowed to be changed by arbitrary multiples of q . It is not hard to see that the vector (f, g) is contained in L^{NT} . For the sample parameter set given in Section 2, the target vector (f, g) has length satisfying

$$\|(f, g)\| \geq \sqrt{1986}.$$

On the other hand, the gaussian heuristic says that the shortest expected vector in L^{NT} has length $\sqrt{Nq/\pi e} \approx 61.34$. Thus the ratio κ given in (6) that gives a measure of how difficult it will be to find the target vector satisfies $\kappa > .72$.

A better attack would be to search for the vector in L^{NT} which is closest to $(0, (1 - 2X - h)p')$, where we choose p' so that $pp' \equiv 1 \pmod{q}$. If successful this would produce F, G such that F is small and $G \equiv Fh - (1 - 2X - h)p' \pmod{q}$ is also small. Then constructing $(1 + pF, 1 - 2X + pG)$ would yield either the original key or a useful substitute. With this approach, after balancing the lattice as in [4] we obtain $\kappa > .23$.

Experimental evidence has shown that if L passes through a sequence of NTRU type lattices of dimension $2N$, $N > 80$, where q, N are related by $q \approx N/2$ and the constant c of (7) is greater than 2.8936, then the extrapolated time necessary for the LLL reduction algorithm to locate either the target vector or a potentially useful substitute is at least T MIPS years, where T is given by the formula

$$\log T = 0.1707N - 15.8184.$$

Thus for $N = 251$ and $c = 2.8936$, one has $T > 5 \cdot 10^{11}$ MIPS-Years.

In this particular instance, $\kappa > .23$ and $N = 251$ corresponds to $c > 5.15$ and so the time required to find the target should be bounded below by 10^{12} MIPS-Years.

B Lifting a NSS Signature Lattice to \mathbb{Z}

An attacker Oscar is presumed to have access to a transcript of signed messages s, s', s'', \dots . This means that he can analyze lists of polynomials

$$f * w, f * w', f * w'', \dots \pmod{q} \quad \text{and} \quad g * w, g * w', g * w'', \dots \pmod{q}. \quad (8)$$

Various ways in which he might try to exploit this mod q information are described in Sections 4.4. In this section we will be concerned with the possibility that Oscar might lift the transcript information (8) and recover the values of $f * w, f * w', \dots$ exactly over \mathbb{Z} .

The reason that this is of concern is that Oscar could then form the lattice L' generated by $X^i * f * w$ for $0 \leq i < N$ and a few different values of w (or similarly for $X^i * g * w$). It is highly likely that the shortest vector in L' is f . Although the lattice L' is not trivial to analyze using lattice reduction, the fact that $\dim(L') = N$, as compared to the NTRU lattice L^{NT} of dimension $2N$, means that L' is easier than L^{NT} . Experimental evidence using LLL and the parameter set given in Section 2 suggests that finding f in L' will take at least 10^4 MIPS-years. If N is increased to 450, the required time increases to approximately 10^{12} MIPS-years. We also note that if Oscar can lift the transcript (8) over \mathbb{Z} , then he can obtain ratios w/w' modulo Q for any Q that he wants. This leads to a lattice of dimension $2N$ that is considerably easier than L^{NT} , although again not trivial. Extensive experiments with this class of lattices indicates that if $N > 680$ then the time required to break the lattice exceeds 10^{12} MIPS years. As our object is to achieve this lower bound with $N = 251$, the existence of these lattices shows that for the parameter set in Section 2, we must explain why it is not possible for Oscar to realistically lift the transcript (8) to \mathbb{Z} .

For any given signature s , Oscar has access to three pieces of information,

$$m, \quad s \equiv f * w \pmod{q}, \quad t \equiv g * w \pmod{q},$$

where the coefficients of s and t are chosen in the range $(-q/2, q/2]$ as usual. He knows that f, g , and w have the form

$$f = f_0 + pf_1, \quad g = g_0 + pg_1, \quad w = m + w_1 + pw_2,$$

where f_0 and g_0 are known, but f_1, g_1, w_1, w_2 are not known. In practice one might take $f_0 = 1$ and $g_0 = 1 - 2X$. For concreteness, we will use these values in our discussion, and to ease notation we write $\bar{m} = g_0 * m$ and $\bar{w}_1 = g_0 * w_1$.

To exploit the transcript and his knowledge of the form of the polynomials, Oscar studies the coefficients at which s and t deviate from m and \bar{m} . We let

$$S = (s - f * w)/q \quad \text{and} \quad T = (t - g * w)/q.$$

The nonzero coefficients of S and T indicate where $f * w$ and $g * w$ wrap modulo q . Multiplying everything out, we can find polynomials A and B so that

$$s = f * w + qS = m + w_1 + pA + qS \quad (9)$$

$$t = g * w + qT = \bar{m} + \bar{w}_1 + pB + qT. \quad (10)$$

By comparing the deviations mod p between s and m , Oscar gains information about the polynomial $w_1 + qS \bmod p$, and similarly for $\bar{w}_1 + qT \bmod p$. Further, since these polynomials both contain w_1 , he may gain useful information by comparing them with one another.

In the following discussion, when we talk about coefficients being the same or different, we always do comparisons modulo p . Now suppose that Oscar observes some index i with $s_i \neq m_i$. From (9), he sees that either $w_{1,i}$ or S_i is nonzero modulo p . If he can reliably distinguish between these two cases, then he will have the information necessary to lift s to \mathbb{Z} (i.e., to find the value of $f * w$ exactly).

A natural tool for separating the two cases is to also look at t_i and t_{i+1} . (The reason both coefficients are of interest is because $g_0 = 1 - 2X$, so a deviation from w_1 appears as consecutive deviations.) If $t_i = \bar{m}_i$ and $t_{i+1} = \bar{m}_{i+1}$, then it is more likely that the s -deviation comes from $S_i \neq 0$; otherwise it is more likely that the s -deviation comes from $w_{1,i} \neq 0$. If w_1 doesn't have very many nonzero coefficients and if it were chosen at random, then this method might enable Oscar to lift to \mathbb{Z} . Of course, he is unlikely to obtain enough information to unambiguously lift, but he might be able to reduce to a manageable number of possibilities.

However, as described in Section 2 (see also Figure 1 in Appendix G), the polynomial w_1 is not chosen at random. Instead, it is chosen to conceal and redistribute the deviations between s , t and m . The basic idea is as follows. First Bob chooses a random w_2 and computes preliminary values of s and t , which we will call

$$s' \equiv f * (m + pw_2) \pmod{q} \quad \text{and} \quad t' \equiv g * (m + pw_2) \pmod{q}.$$

For each index $0 \leq i < N$, Bob checks if s'_i and/or t'_i differ from m_i (always comparing modulo 3), and if so, whether they differ in the same way or in a different way. Then Bob chooses $w_{1,i}$ according to this comparison so as to conceal deviations. For example, if $s'_i = t'_i \neq m_i$, then taking $w_i = m_i - s'_i$ will cause the s deviation to disappear and the t deviation to disappear from the i th coefficient. When we say the deviations “disappear,” we mean that they will no longer be visible to Oscar when Bob computes the final value of s .

Similarly, if s and t both have deviations, but they are not equal, then Bob randomly chooses the coefficient of w_1 to cancel the deviation in one or the other of them. Finally, if s_i or t_i , but not both, gives a deviation, then a certain proportion of the time (e.g., 25%) Bob puts a coefficient into w_1 to cancel the deviation, which has the effect of moving it into the other polynomial and/or into another coefficient of the same polynomial.

The net effect is that as long as there are a reasonably large number of deviations caused by S and T and also a significant number of simultaneous deviations from S and T , then Oscar will have no way to use the visible deviations to reconstruct $f * w$ and/or $g * w$ exactly over \mathbb{Z} . Table 6 in Appendix H gives results of experiments using the parameters from Section 2. Thus for example, a typical signature will have at least six simultaneous (S, T) deviations that are effectively hidden from Oscar by w_1 . If Oscar tries to guess where they are, he is faced with $\binom{251}{6} \approx 2^{38}$ choices, and this ignores the fact that he also has to guess the direction of the deviation and the fact that these simultaneous deviations are only some of the ambiguity that he needs to resolve in order to lift s or t . Further, Bob needs to lift at least two signatures before he obtains a usable lattice. We also note that Bob can precompute the number of deviations that are being moved or hidden by w_1 and, if he occasionally feels that there are too few, he can simply discard that s and begin again with a new random w_2 .

We will mention one additional approach that an attacker could take. Given $s \equiv f * w \equiv (f_0 + pf_1) * w \pmod{q}$ and $t \equiv g * w \equiv (g_0 + pg_1) * w \pmod{q}$, one could compute $p'(g_0 * s - f_0 * t) \pmod{q}$, where $p'p \equiv 1 \pmod{q}$. (Recall that f_0, g_0 are public.) If $(g_0 * f_1 - f_0 * g_1) * w$ had no reduction modulo q in its coefficients, then after translation to the interval $[-q/2, q/2 - 1]$ the polynomial $(g_0 * f_1 - f_0 * g_1) * w$ would actually be recovered exactly over \mathbb{Z} . A lattice attack such as those described in above could then be launched against several of these products and w could probably be recovered in about 10^2 MIPS years. However, with the choices $f_0 = 1$ and $g_0 = 1 - 2X$ as described in Section 2, there is always a considerable amount of reduction modulo q . Indeed, we find that there are more than 10^{20} possible liftings back to \mathbb{Z} for each such product, and a minimum of two correct liftings would be required to begin a lattice attack.

In our discussion of lifting, one other point should be made. If Oscar looks at s modulo q and marks the coefficients that deviate modulo p from m , he might suspect that the coefficients that are close to the edges (i.e., near to $-q/2$ or $q/2$) are more likely to come from wrapping. In other words, the product $f * w$ has coefficients outside the range $(-q/2, q/2]$ that

are wrapped, and Oscar might hope that they just wrap a little bit. It turns out that this is true to some extent. The wrapped coefficients are a little more likely to appear at the edges; but in general there will be many wrapped coefficients that stretch well away from the edges, so Oscar cannot hope to lift s to \mathbb{Z} by simply altering the coefficients near the edges. For the parameter set described in Section 2, Table 5 in Appendix H gives the distribution of the values of the coefficients that wrap.

C Forgery Via Lattice Reduction

The opponent, Oscar, can try to forge a signature s on a given message m by means of lattice reduction. The best method we have found for accomplishing this is described in the following section. It requires the successful analysis of a lattice of dimension $3N$ and appears to have a time requirement in excess of the 10^{12} MIPS years we have mentioned previously.

In order to analyze this attack, we define the Sup Norm (also called the L^∞ norm) of a polynomial a to be the quantity

$$\|a\|_\infty = \max\{|a_0|, |a_1|, \dots, |a_{N-1}|\},$$

where remember that a polynomial modulo q has its coefficients reduced into the range between $-q/2$ and $q/2$.

Given a message m , Oscar needs to construct a signature s that satisfies the following properties.

- (A) $s = m_1 + ps_1$, where m_1 differs from $f_0 * m \bmod p$ in at least D_{\min} and at most D_{\max} places and $\|s_1\|_\infty < q/(2p)$
- (B) Given $t \equiv h * s \pmod{q}$, t must also have the form $t = m_2 + pt_1$, where m_2 differs from $g_0 * m \bmod p$ in at least D_{\min} and at most D_{\max} places and $\|t_1\|_\infty < q/(2p)$.

The optimal approach for Oscar seems to be to search for s_1, t_1 simultaneously by constructing the following lattice:

$$L_F = \begin{bmatrix} \alpha I & 0 & ph \\ 0 & \beta I & -pI \\ 0 & 0 & qI \end{bmatrix}$$

The variables α, β are weights that Oscar will choose to maximize his chances of success. Let $m_1 = f_0 * m + \epsilon_1$, where ϵ_1 with small norm will be determined shortly. Suppose now that Oscar could find a point in L_F

that is very close to the point $(0, 0, g_0 * m - h * m_1 \pmod q)$. (Here each entry stands for N entries, and the right hand entry has been reduced modulo q so that every coefficient lies between $-q/2$ and $q/2$.) This point, or target vector would then have the form $(\alpha s'_1, \beta t'_1, phs'_1 - pt'_1 \pmod q)$. If it were close to $(0, 0, g_0 * m - h * m_1 \pmod q)$ then

$$\|(\alpha s'_1, \beta t'_1, phs'_1 - pt'_1 - g_0 * m + h * m_1 \pmod q)\|$$

would be small. Denote this difference by

$$\epsilon_2 \equiv phs'_1 - pt'_1 - g_0 * m + h * m_1 \pmod q,$$

so that $\|\epsilon_2\|$ is small. Then a reasonable strategy for Oscar would be to set $s = m_1 + ps'_1$ and $t = m_2 + pt'_1$, where m_1 was given above and $m_2 = g_0 * m + \epsilon_2$. Oscar would require ϵ_1, ϵ_2 to have Hamming weights lying between D_{\min} and D_{\max} . This would have the greatest chance of producing a valid signature if α, β were chosen so that the three groups of N coordinates are weighted equally. Oscar also needs

$$\|s'_1\|_\infty < q/(2p), \quad \|t'_1\|_\infty < q/(2p), \quad \text{and} \quad \|\epsilon_2\| < \sqrt{D_{\max}}.$$

In general, if \mathbf{v} is a vector of dimension N whose coordinates are more-or-less randomly distributed in some interval $[-r, r]$, then

$$\|\mathbf{v}\| \approx \sqrt{N/3} \|\mathbf{v}\|_\infty.$$

So the above L_∞ norms on s'_1, t'_1 will probably be satisfied if

$$\|s'_1\| \leq \sqrt{N/3} \cdot q/2p \quad \text{and} \quad \|t'_1\| \leq \sqrt{N/3} \cdot q/2p.$$

Oscar should then set $\alpha = \beta$ and $\alpha q/(2p) \sqrt{N/3} = \sqrt{D_{\max}}$. Substituting into (6) it can then be checked that for the choice of parameters in Section 2, the point in L_F that Oscar must locate is a factor of

$$\kappa = 1.805q^{1/3}(D_{\max}/N)^{1/6}p^{-2/3} = 3.637$$

times the shortest expected length of a vector in L_F away from the point $(0, 0, g_0 * m - h * m_1 \pmod q)$. Thus for these parameters Oscar must solve the closest vector problem for a point less than 4 times the length of the shortest vector away from a given point, in a lattice of dimension 753. Experiments indicate that this should require more than the lower bound of 10^{12} MIPS-years.

Note, however, that there is one point in L_F that Oscar can locate very quickly, namely 0. If 0 were an allowable signature, Oscar could

set $s'_1 = 0$ and $\epsilon_1 = -f_0 * m$, so $m_1 = s = 0$. Then $s = 0$ would be an allowable signature on m (and equivalently, the origin would be sufficiently close to the point $(0, 0, g_0 * m - h * m_1 \pmod{q})$) if $D_{\min} \leq \text{Dev}(0, f_0 * m), \text{Dev}(0, g_0 * m) \leq D_{\max}$. This is why $s = 0$ is excluded from the collection of valid signatures.

D Transcript Averaging Attacks

As mentioned previously, examination of a transcript (4) of genuine signatures gives the attacker a sequence of polynomials of the form

$$s \equiv f * w \equiv (f_0 + pf_1)(m + w_1 + p * w_2) \pmod{q}$$

with varying w_1 and w_2 . A similar sequence is known for g . Because of the inherent linearity of these expressions, we must prevent Oscar from obtaining useful information via a clever averaging of long transcripts.

The primary tool for exploiting such averages is the *reversal* $\rho(a)$ of a polynomial $a(X) = \sum a_i X^i$ defined by

$$\rho(a) = a(X^{N-1}) = a_0 + a_{N-1}X + a_{N-2}X^2 + \cdots + a_1X^{N-1}.$$

The function ρ satisfies $\rho(a + b) = \rho(a) + \rho(b)$ and $\rho(ab) = \rho(a)\rho(b)$; it is a homomorphism $\rho : R \rightarrow R$. A calculation shows that the average of $a\rho(a)$ over a long list of random a with uncorrelated coefficients will approach a constant polynomial whose value is the limiting average of $\|a\|^2$. On the other hand, the average $a\rho(a')$ over a list of uncorrelated polynomials a and a' will approach 0.

For each s , Oscar has access to m and to $m' \equiv s \pmod{q}$. Note that $f_0 * m$ and m' will be very similar, differing by at most D_{\max} coefficients. Suppose that for each s , Oscar chooses some M , equal to $f_0 * m$ or m' or some linear combination of these polynomials. He can then compute the average of $\rho(M)s$ over many signatures constructed with the same f .

Although s is not equal to $f * w$, it differs from $f * w$ by a polynomial qE . In other words, we can write

$$s = f * (m + w_1 + p * w_2) + q * E,$$

where E has a ± 1 at every coefficient where a nontrivial reduction mod q occurs in $f * w$ and zeros elsewhere. (There could be a few ± 2 coefficients in E as well.) We will write $(a)_\infty$ for the limiting value of the average of a collection of polynomials $\{a\}$. Then assuming that E and M are uncorrelated, we find that

$$(\rho(M) * s)_\infty = f * (\rho(M) * (m + w_1 + p * w_2))_\infty \quad (11)$$

Suppose that we take $M = m$ and that w_1 and w_2 are uncorrelated with m . Then the righthand side of (11) equals $f * (\rho(m)m)_\infty$ and the key f is revealed.

Fortunately, there is an easy way to defeat this and other similar averaging attacks. We simply choose w_2 so that it is partially correlated with m and w_1 . We want to do this in such a way that $(\rho(M) * (m + w_1 + p * w_2))_\infty = 0$. Note that in any particular message, we cannot use w_2 to “cancel” m and w_1 , but it is only necessary that this cancellation occurs on average. Thus when we constructing a signature, we begin by choosing a preliminary w_1 and w_2 . Then for each $0 \leq i < N$ we flip a p -sided coin and with probability $1/p$, we replace $w_{2,i}$ (i.e., the coefficient of X^i in w_2) with $w_{2,i} - m_i - w_{1,i}$. (For the full algorithm used to choose w_1 and w_2 , see Section 2 and Figure 1 in Appendix G.) With this choice of w_2 , the limiting value (11) is 0 for any linear combination M of m and w_1 . Similar remarks apply to transcripts of signatures t constructed from g .

Oscar can formulate a similar attack by selecting a subset of a transcript containing signatures whose associated messages can be rotated so that they have a particular bit pattern in common. (For example, signatures rotated so that the first bit of the message is always 1.) Then an average over this subset will eventually converge to a constant multiple of the key f . However, with the w_2 construction described above, the constant will again equal zero.

We conducted numerical experiments with w_2 constructed as above. An examination of transcripts formed with 10 million signatures revealed no discernable correlation between the limiting average and the key f .

Oscar can also try to average $s * \rho(s)$, which should yield a constant times $f * \rho(f)$ in the limit. As discussed in [5, 6], knowledge of this product does not seem to reveal any useful information about f . Further, experiments indicate that even a transcript of 10^7 signatures is not sufficient to determine the value of $f * \rho(f)$.

Finally, we mention that Coppersmith has described a fourth moment attack (see [5, 6]) in which one computes the limiting value of products of four polynomial coefficients of s (or t). It is likely that if one could compute a fourth moment limit of this sort, then one would obtain some information about f and g . However, since we have noted that 10^7 signatures is insufficient to obtain the limiting value of a second moment, and the fourth moment will settle down to a limiting value considerably more slowly, it appears that a fourth moment attack is impractical.

E A Probabilistic Argument for Soundness

The argument for soundness is based upon the heuristic described in Appendix A.1. Suppose that Oscar can forge a signature s on a message m . This signature will correspond to a point $(\alpha s, \alpha h * s, s, h * s)$ in a $4N$ dimensional lattice L_S . Here α is an appropriate weight, the second group of N coordinates is adjusted modulo αq , the third modulo 3, and the fourth modulo both q and 3. The determinant of L_S equals $(9\alpha^2 q)^N$. A valid signature will be close to the point $(0, 0, f_0 * m \pmod{3}, g_0 * m \pmod{3})$. If we set $\alpha = (2/q)\sqrt{3D/N}$ then the lattice is balanced and the signature point, appropriately reduced modulo q and 3 is within a factor of $\kappa = 1.28(D_{\max}q/N)^{1/4}$ times the expected shortest vector in L_S away from $(0, 0, f_0 * m \pmod{3}, g_0 * m \pmod{3})$. For any fixed D_{\max} and q , this ratio κ goes below 1 as N increases. We know that such a point exists as we can use f to construct such an s . The fact that the ratio is less than 1 indicates that the probability of such a lattice point existing in L_S by chance, i.e., not by the specific construction using f , approaches 0.

Assume therefore, that for a given m Oscar can create a signature of the form $f * w$, where $w = m + w_1 + pw_2$. If he does this for pairs of $m, m + 1$ (ignoring the restriction on precise numbers of 1's and -1 's in m), he will be able to create differences of the form $f * (1 + w')$. Only a moderate number of such products would be needed to obtain f by averaging.

F Comparison of NSS and NTRU

We briefly describe the NTRU public key cryptosystem and compare it to NSS. In both systems the two numbers p and q are chosen to be relatively prime to one another and q is considerably larger than p , although even q is not large. A typical NTRU parameter set is $(N, p, q) = (263, 3, 128)$.

Bob creates his secret NTRU key by choosing two polynomials f and g modulo p , where f has the form $f = 1 + pf_1$ and there may be some further restriction on how many nonzero coefficients are allowed. Bob computes the inverse f^{-1} for f modulo q . (In the original description of NTRU, f did not have this special form and it was necessary to also compute and use the inverse of f modulo p . Taking $f = 1 + pf_1$ increases efficiency without affecting the security of the original scheme.) Bob then forms his public NTRU encryption key

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

The polynomial f is Bob's private NTRU decryption key. When Alice wants to send a message to Bob, she chooses a polynomial m modulo p as her plaintext, she chooses a random polynomial r modulo p , and she sends to Bob the ciphertext

$$e \equiv pr * h + m \pmod{q}.$$

Bob first computes $a \equiv f * e \pmod{q}$, where he chooses the coefficients in a certain specific interval of length q , and then he recovers the plaintext by computing $a \pmod{p}$. See [4, 11] for an explanation of why this decryption process works, a security analysis, and suggested values for the parameters.

Notice that the NTRU key and the NSS key both have the form $h = f^{-1} * g \pmod{q}$. In each case, the system is broken if an attacker can find a small polynomial F (i.e., a polynomial with small coefficients) with the property that $F * h \pmod{q}$ is also small. The problem of finding such an F is naturally formulated in terms of finding a small vector in a lattice. Thus underlying both NTRU and NSS is the problem of lattice reduction. See Section 4.3 for details on how lattice attacks are formulated and estimates for the time they require.

G Algorithm for Selecting w_1 and w_2

The precise algorithm used for selecting w_1 and w_2 is described using C pseudocode in Figure 1.

H Numerical Tables

This appendix contains tables describing the results of numerical experiments that are referred to in the text of this article.

```

/* Algorithm to Construct w1 and w2 */
N = 251; q = 128; p = 3;
w1Limit = 25; SwitchProb = 0.25;
Choose w2[] randomly with 32 1's and 32 -1's
Set w1[] = 0;
Compute s = f*(m+p*w2) and t = g*(m+p*w2)
Reduce the coordinates of s and t modulo q into the range (-q/2,q/2]
Reduce the coordinates of s and t modulo p into the range (-p/2,p/2]
for ( i = 0; i < N; i++ ) {
    if ( s[i] != m[i] && t[i] != m[i] && s[i] == t[i] )
        w1[i] = m[i] - s[i] mod p;
    if ( s[i] != m[i] && t[i] != m[i] && s[i] != t[i] )
        w1[i] = 1 or -1 chosen randomly;
}
for ( i = 0; i < N; i++ ) {
    if ( s[i] != m[i] && t[i] == m[i] )
        w1[i] = m[i] - s[i] mod p with probability SwitchProb;
    if ( t[i] == m[i] && t[i] != m[i] )
        w1[i] = m[i] - t[i] mod p with probability SwitchProb;
    if ( w1[i] has more than w1Limit nonzero coordinates )
        break out of the loop;
}
for ( i = 0; i < N; i++ )
    With probability 1/p, set w2[i] = w2[i] - (m[i] + w1[i]);

```

Fig. 1. Algorithm for the Construction of w_1 and w_2

Range	Dev($s, f_0 * m$)	Dev($t, g_0 * m$)
32 to 39	0.02%	0.08%
40 to 47	0.38%	0.99%
48 to 55	3.53%	6.98%
56 to 63	14.21%	26.32%
64 to 71	27.58%	37.79%
72 to 79	28.51%	21.22%
80 to 87	17.03%	5.58%
88 to 95	6.54%	0.90%
96 to 103	1.74%	0.11%
104 to 158	0.46	0.02%

$(N, p, q) = (251, 3, 128)$ — 10^6 Trials

Table 2. Number of Deviations Between m and s and t

D_{\min}	D_{\max}	Probability
55	82	$2^{-90.86}$
55	87	$2^{-80.95}$
55	92	$2^{-71.66}$
55	98	$2^{-61.32}$

Table 3. Probability Random t Satisfies $D_{\min} \leq \text{Dev}(t, g_0 * m) \leq D_{\max}$

Number Wrapped	in $f * w$
36 to 51	1.67%
52 to 55	2.58%
56 to 59	4.88%
60 to 63	7.91%
64 to 67	11.00%
68 to 71	13.19%
72 to 75	13.89%
76 to 79	12.83%
80 to 83	10.65%
84 to 87	8.02%
88 to 91	5.50%
92 to 95	3.45%
96 to 99	2.05%
100 to 175	2.36%

Number Wrapped	in $g * w$
12 to 19	0.26%
20 to 23	1.09%
24 to 27	3.40%
28 to 31	7.35%
32 to 35	12.08%
36 to 39	15.68%
40 to 43	16.48%
44 to 47	14.72%
48 to 51	11.26%
52 to 55	7.64%
56 to 59	4.68%
60 to 63	2.62%
64 to 67	1.38%
68 to 175	1.36%

Coefs of $f * w$ Outside $[-q/2, q/2]$

Coefs of $g * w$ Outside $[-q/2, q/2]$

Table 4. Number of Wrapped Coefficients— $(N, p, q) = (251, 3, 128)$ — 10^6 Trials

Coefficient Range	Moved in s_i	Moved in t_i
$-64 \leq s_i, t_i \leq -57$	8.64%	13.26%
$-56 \leq s_i, t_i \leq -49$	9.02%	11.25%
$-48 \leq s_i, t_i \leq -41$	7.78%	8.44%
$-40 \leq s_i, t_i \leq -33$	5.69%	5.69%
$-32 \leq s_i, t_i \leq -25$	5.69%	4.43%
$-24 \leq s_i, t_i \leq -17$	4.95%	3.21%
$-16 \leq s_i, t_i \leq -9$	4.10%	2.29%
$-8 \leq s_i, t_i \leq -1$	4.23%	2.01%
$0 \leq s_i, t_i \leq 7$	3.96%	1.93%
$8 \leq s_i, t_i \leq 15$	4.34%	2.33%
$16 \leq s_i, t_i \leq 23$	4.90%	3.08%
$24 \leq s_i, t_i \leq 31$	4.90%	3.97%
$32 \leq s_i, t_i \leq 39$	6.51%	5.94%
$40 \leq s_i, t_i \leq 47$	7.59%	8.11%
$48 \leq s_i, t_i \leq 55$	7.43%	9.92%
$56 \leq s_i, t_i \leq 63$	10.27%	14.14%

$(N, p, q) = (251, 3, 128)$ — 10^6 Trials

Table 5. Wrapped Coefficients of $s \equiv f * w \pmod{q}$ and $t \equiv g * w \pmod{q}$

Number Wrapped	Same Way	Opposite Way
0 to 2	7.80%	8.90%
3 to 5	31.10%	33.80%
6 to 8	34.90%	31.50%
9 to 11	18.40%	16.70%
12 to 14	5.90%	6.70%
15 to 26	1.90%	2.40%

$(N, p, q) = (251, 3, 128)$ — 10^3 Trials

Table 6. Coefficients of $f * w$ and $g * w$ Simultaneously Outside $[-q/2, q/2]$

$\text{Dev}(s, m') + \text{Dev}(t, g_0 * m')$	Numerical Data	Normal Distribution
232 to 239	0.01%	0.01%
240 to 247	0.11%	0.10%
248 to 255	0.77%	0.74%
256 to 263	3.58%	3.55%
264 to 271	10.78%	10.86%
272 to 279	21.16%	21.28%
280 to 287	26.69%	26.67%
288 to 295	21.53%	21.38%
296 to 303	11.02%	10.96%
304 to 311	3.55%	3.59%
312 to 319	0.71%	0.75%
320 to 327	0.09%	0.10%
328 to 335	0.01%	0.01%

$(N, p, q) = (251, 3, 128)$ — 10^6 Trials

Table 7. Number of Deviations for Fixed (s, t) and Randomly Chosen m'

References

1. E.F. Brickell and K.S. McCurley. *Interactive Identification and Digital Signatures*, AT&T Technical Journal, November/December, 1991, 73–86.
2. 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.
3. J. Hoffstein, B.S. Kaliski, D. Lieman, M.J.B. Robshaw, Y.L. Yin, *A New Identification Scheme Based on Polynomial Evaluation*, patent application.
4. 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.
5. 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.
6. 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.
7. A.J. Menezes, *Software Implementation of Elliptic Curve Cryptosystems Over Binary Fields*, presentation at CHES 2000, August 17, 2000.
8. A.J. Menezes and P.C. van Oorschot and S.A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1996.
9. 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.

10. 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.
11. J.H. Silverman. Estimated Breaking Times for NTRU Lattices, NTRU Technical Note #012, March 1999, <www.ntru.com>.
12. J.H. Silverman. Almost Inverses and Fast NTRU Key Creation, NTRU Technical Note #014, March 1999, <www.ntru.com>.
13. 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.
14. 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.
15. D. Stinson, *Cryptography: Theory and Practice*. CRC Press, 1997.