Cryptanalysis of the NTRU Signature Scheme (NSS) from Eurocrypt 2001

Craig Gentry[1], Jakob Jonsson[2], Jacques Stern[3]⋆, and Michael Szydlo[2]

[1] DoCoMo Communications Laboratories USA, Inc.
cgentry@dcl.docomo-usa.com
[2] RSA Laboratories, 20 Crosby Drive, Bedford, MA 01730, USA
{jjonsson,mszydlo}@rsasecurity.com
[3] Dépt d'Informatique, Ecole normale Supérieure, Paris, France
Jacques.Stern@ens.fr

Keywords: NSS, NTRU, Signature Scheme, Forgery, Transcript Analysis, Lattice, Cryptanalysis, Key Recovery, Cyclotomic Integer.

Abstract. In 1996, a new cryptosystem called NTRU was introduced, related to the hardness of finding short vectors in specific lattices. At Eurocrypt 2001, the NTRU Signature Scheme (NSS), a signature scheme apparently related to the same hard problem, was proposed. In this paper, we show that the problem on which NSS relies is much easier than anticipated, and we describe an attack that allows efficient forgery of a signature on any message. Additionally, we demonstrate that a transcript of signatures leaks information about the secret key: using a correlation attack, it is possible to recover the key from a few tens of thousands of signatures. The attacks apply to the recently proposed parameter sets NSS251-3-SHA1-1, NSS347-3-SHA1-1, and NSS503-3-SHA1-1 in [2]. Following the attacks, NTRU researchers have investigated enhanced encoding/verification methods in [11].

1 Introduction

Recently, Hoffstein, Pipher, and Silverman introduced a public-key signature scheme called NSS (the "NTRU Signature Scheme") [9]. This scheme is related to the NTRU cryptosystem, which was first introduced at the CRYPTO '96 rump session. An attack on NTRU was quickly found by Coppersmith and Shamir (see [4]), which led the authors to adopt larger parameters, and reformulate the underlying hard problem as a lattice problem. The current version of NTRU, as published in [6], remains unbroken. NSS is also related to the problem of finding short vectors in certain lattices, and is an improvement over an early version [7] presented at the CRYPTO 2000 rump session. This version proved to be insecure, which the designers observed at an early stage. Ilya Mironov [14] made the same observation independently a few months later. Basically, it appeared that signatures leaked information about the private key, which allowed for statistical attacks.

To eliminate the disclosed weaknesses, certain adaptations were made, yielding the scheme described in [9] and [8]. Unfortunately, these signatures *still* leak information about the private key. More precisely, it turns out that correlations between certain coefficients in the signature and the private key are sufficient to recover the entire public key.

Moreover, and even more dramatic, is a direct forgery attack which enables an adversary to sign arbitrary messages without any knowledge of the private key. While the flaw does not rule out potentially secure future revisions, our analysis shows that the scheme as presented in [9], [8] and [2] is completely insecure.

This paper is organized as follows. In section 2 we provide background and describe NSS in more detail. In section 3 we describe the efficient forgery procedure. Next, in section 4 we explain how to recover the key by examining valid signatures. In section 5 we discuss some revisions suggested by the authors of NSS to repair the signature scheme.

⋆ This work has been partially supported by the French Ministry of Research under the RNRT Project "Turbo-Signatures"

2 Description of NSS

Here we review some mathematics that underlie NSS, and give a brief description of the scheme. We refer readers to [8] and [9] for more detailed information.

2.1 Background Mathematics

The key underlying mathematical structure of the scheme is the polynomial ring

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

where N and q are integers. In practice, N is prime (e.g., 251) and q is a power of 2 (e.g., 128). Elements in R are polynomials of degree (at most) $N-1$ and with coefficients in the range $(-q/2, q/2]$.

Multiplication in this ring is like ordinary polynomial multiplication, but subject to the relations $X^{N+k} = X^k$ for any $k \geq 0$. This means that the coefficient of X^k in the product $a * b$ of $a = a_0 + a_1 X + \ldots + a_{N-1}X^{N-1}$ and $b = b_0 + b_1 X + \ldots + b_{N-1}X^{N-1}$ is

$$(a * b)_k = \sum_{i+j=k \bmod N} a_i b_j. \tag{2}$$

The multiplication of two polynomials in R is also called the *convolution product* of the two polynomials. For any polynomial $a \in R$, it is also convenient to introduce the *convolution matrix* of a as follows: Let M_a be the $N \times N$ matrix indexed by $\{0, \ldots, N-1\}$, where the element on position (i,j) is equal to $a_{(j-i) \bmod N}$. With this representation, the product of a and b can be also expressed as the product of the row vector $(a_0, \ldots, a_{N-1})$ with the matrix M_b. ¿From now on, we will freely identify any polynomial with its corresponding row vector.

While the ring (1) may seem unnatural at first, it is directly related to the ring of integers in the cyclotomic field

$$\mathbb{Q}(\zeta_N) = \mathbb{Q}[X]/(X^{N-1} + \ldots + X + 1). \tag{3}$$

This field $\mathbb{Q}(\zeta_N)$ is a field extention of the rational numbers $\mathbb{Q}$, and has a subring of *Algebraic Integers*, $\mathbb{Z}(\zeta_N)$, analogous to the ordinary integers $\mathbb{Z} \subset \mathbb{Q}$. In fact, the set of polynomials $p \in R$ with $p(1) = 1$ is isomorphic to the integers $\mathbb{Z}(\zeta_N) \subset \mathbb{Q}(\zeta_N)$, and the convolution product described above in (2) is simply the ordinary multiplication operation in this field.

This field has been extensively studied and has been proposed for use in other cryptographic applications such as factoring and as basis for a public key cryptosystem (e.g. [17]), and is likely to appear in further analysis of NTRU related cryptosystems. However, further familiarity with this field is not required for the rest of this paper.

2.2 The NSS Signature Scheme

The public key of NSS consists of a polynomial h of degree $N-1$, and the private key of the scheme consists of two polynomials f and g with "small coefficients" such that $f * h = g$, where the polynomials are elements of $R = \mathbb{Z}_q[X]/(X^N - 1)$, and q and N are typically 128 and 251.

In order to describe the scheme further, additional parameters are needed. These parameters include the integer p, which is typically chosen to be 3, and the integers d_f, d_g and d_m, whose suggested values are respectively 70, 40 and 32. The latter parameters are used to define several families of polynomials denoted by $\mathcal{L}(d_1, d_2)$, a notation that refers to the set of polynomials of degree at most $N-1$ with d_1 coefficients 1, d_2 coefficients -1 and all other coefficients 0.

Key generation: Two polynomials f and g are defined as

$$f = f_0 + pf_1$$

$$g = g_0 + pg_1$$

where f_0 and g_0 are publicly known small polynomials (typically $f_0 = 1$ and $g_0 = 1 - 2X$). The polynomial f_1 is randomly chosen from $\mathcal{L}(d_f, d_f)$ and similarly g_1 is randomly chosen from $\mathcal{L}(d_g, d_g)$. It is required that f be invertible (i.e., there exists some f^{-1} with $f * f^{-1} = 1 \bmod q$). This is true with very high probability; in any case the preceding step may be repeated by choosing a different polynomial f_1.

Signature generation: To sign a message, one transforms the message to be signed into a message representative according to a hash function-based procedure such as that described in [2]. This message representative is a polynomial in $\mathcal{L}(d_m, d_m)$. The signer first computes

$$w = m + w_1 + pw_2$$

where w_1, w_2 are two polynomials with small coefficients generated at random in a rather complex manner that is described in Appendix B. The signer next computes the convolution

$$s = f * w \bmod q$$

and outputs the pair (m, s) as the signature of m.

Signature verification: A signature (m, s) consists of the message m together with the polynomial s of degree $N-1$, with coefficients reduced modulo q. Signature verification depends on two further parameters $D_{\min}$ and $D_{\max}$ (paper [9] suggests $D_{\min} = 55$ and $D_{\max} = 87$, together with the parameters suggested above), and upon the concept of *Deviation*. Given two polynomials, A and B of degree $N - 1$, the deviation $\mathrm{Dev}(A, B)$ is the function that counts the number of coefficients where $(A \bmod q) \bmod p$ and $(B \bmod q) \bmod p$ differ. Here, modular reduction computes the coefficients in the interval $(-q/2, q/2]$ (resp. $(-p/2, p/2]$). If A and B are two random polynomials in the ring $\mathbb{Z}_q[X]/(X^N - 1)$ and p equals 3, we would expect $\mathrm{Dev}(A, B)$ to be about $\frac{2}{3}N \approx 167$, since the probability that A_i and B_i differ modulo 3 is about $\frac{2}{3}$.

To verify a signature, first it is checked that $s \neq 0$. Then the polynomial $t = s * h \pmod q$ is computed, and the two conditions

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

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

are checked. If both conditions hold, the signature is accepted as valid.

The soundness of the scheme follows from technical estimates, which we omit. It should be noted that signature generation does not necessarily produce valid signatures. With the above parameters, signature verification fails in twenty percent of the cases and, when this happens, the signer has to create another signature.

3 Forgery Attacks

Paper [9] claims that a signature essentially proves possession of the secret trapdoor. Further, it envisions several potential attacks and concludes that the security of the system, with the above parameters, is comparable to RSA with 1024 bit moduli. We show that an attacker can generate forgeries (with slightly fewer than $D_{\max} = 87$ deviations) almost as quickly as the signer can generate signatures, without any knowledge of the private key. Furthermore, the attacker can generate forgeries with substantially fewer than $D_{\max}$ deviations by using lattice reduction.

3.1 Basic Forgery Attack: The Principle

In [9] and [8], NSS and NTRU are described as being based on essentially the same hard lattice problem. In fact, the problem underlying NSS is more of an error correction problem and, as demonstrated in many papers (see e.g. [16]), such problems take much larger dimensions to become hard.

The attack is very simple, once the perspective has been changed, as just indicated. The attacker's task is to find a pair of polynomials (s, t) that satisfy $t = s * h \pmod q$, as well as the deviation requirements:

$$55 \le \mathrm{Dev}(s, f_0 * m) \le 87;$$

$$55 \le \mathrm{Dev}(t, g_0 * m) \le 87.$$

Since s and t have $2N$ coefficients altogether, and the equation $t = s * h \pmod q$ imposes N linear constraints, the attacker has N degrees of freedom remaining in s and t with which he can try to satisfy the deviation requirements. With these N degrees of freedom, he sets

$$s_i \equiv (f_0 * m)_i \bmod p$$

and

$$t_j \equiv (g_0 * m)_j \bmod p$$

for $\lfloor N/2 \rfloor$ coefficients of s and $\lceil N/2 \rceil$ coefficients of t — i.e., he chooses about half the coefficients of s and half of t to be non-deviating. The remaining halves of s and t are left to chance. Since the chosen half of s (resp. t) has no deviations, and the remaining half will probabilistically deviate in about $\frac{2}{3}$ of the positions, overall about $\frac{1}{3}$ of the coefficients of s (resp. t) will deviate. Since $\frac{1}{3}N \approx 84 \le D_{\max}$ for $(N, D_{\max}) = (251, 87)$, this process will usually generate a valid forgery after only a few iterations. In general, if $p = 3$ and $D_{\max} \ge \frac{1}{3}N$, then this attack will generate forgeries regardless of the size of N.

3.2 Basic Forgery Attack: The Details

In practice, the attack is slightly more complicated than the above, because it is possible that the constraints on s and t are incompatible. In this case, we say the attacker is *unlucky*. To avoid being unlucky, the attacker constrains only $k < N/2$ coefficients each of s and t. By setting up linear equations based on the constraints on t, we obtain a system of k linear equations modulo q over the $(N - k)$ free unknowns. The coefficients of the unknowns in this system form a $k \times (N - k)$ submatrix M of M_h whose coefficients are modulo q integers. We make the heuristic assumption that these coefficients are independent random bits, when reduced modulo 2.

Lemma 1. *Based on the heuristic assumption, the attacker is unlucky with probability at most* $\epsilon = \frac{1}{2^{N-2k}}$

Proof of lemma: We show that, with probability at least $1 - \epsilon$, the columns of $M \bmod 2$ generate the entire k-dimensional space over the two-element field. If this holds, the system has rank k and it has solutions modulo 2 and modulo q as well, since q is a power of 2. Now, for every k-bit vector x, a column vector v is such that the inner product (v, x) is zero with probability 1/2. Since there are $N - k$ independent columns, x is orthogonal to all column vectors with probability $\frac{1}{2^{N-k}}$. Since there are 2^k possible values for x, we get that, with probability at least $1 - \frac{1}{2^{N-2k}}$, there is no vector orthogonal to all column vectors of $M \bmod 2$. This means that these column vectors span the entire space. □

Setting $k = 121$, the attacker will be lucky with probability at least $1 - 2^{-9}$. Assuming he is lucky, the attack now amounts to solving a system of 121 equations with 130 unknowns. However, a closer look shows that the matrix corresponding to this system does not depend on m, provided the attacker keeps the same selection of coordinates for his constraints; only the "righthand side" of the linear system does. This makes possible standard preprocessing of the linear system. To keep things simple, assume that, by suitably reindexing coefficients, one has brought the constrained coefficients of s in front and made the constrained coefficients of t the trailing block. Then, the matrix M of the system that the attacker has to solve is at the right bottom corner of M_h, defined by the last k rows of M_h and its last $N-k$ columns. Further relabeling makes the last k columns of M an invertible submatrix U. Again U lies at the bottom right corner of M_h. Thus, once the attacker precomputes the inverse U^{-1} of U, he may thereafter generate solutions to the linear system by choosing the $N-2k=9$ middle coordinates of s arbitrarily and obtaining the k last ones by a single multiplication by U^{-1}. For $k=121$, one readily checks that the obtained solution will satisfy the deviations requirement with probability $\geq 1/4$, so the attacker can expect to obtain the desired forgery after only 4 such multiplications. This makes forgery almost as fast as regular signature generation.

Alternatively, the attacker may search for a solution whose number of deviations lies closer to the middle of the interval $(D_{\min}, D_{\max})$, simply by searching through the 128^9 solutions to his linear equations. In a relatively short time, he can expect to find a solution (s,t) for which s and t have, for example, only 75 deviations.

A computer program written in C confirms the above analysis. Specifically, we have carried out the following two experiments:

1. The first with a public key that we manufactured, corresponding to the parameters from [9].
2. The second with a public key coming from one challenge from the NTRU web site. This challenge is for the encryption scheme. Unfortunately, there is no challenge for the signature scheme, but we wished to make it clear that we were working without the secret key. The challenge uses $N=263$ instead of $N=251$. We left the other parameters unchanged and observe that raising N only makes the forgery slightly more difficult.

We found forgeries whose distance pairs are respectively $(75,74)$ and $(79,79)$, close to the middle of the interval $(D_{\min}, D_{\max})$.

3.3 Forgery Attack with Lattice Reduction

In this section we make use of *lattice reduction*, a technique to find useful $\mathbb{Z}$ bases of lattices (discrete subgroups of $\mathbb{R}^n$). The celebrated LLL algorithm [13] is one of a family of algorithms that find bases containing short vectors in a lattice, and has found many uses in cryptology. The contemporary survey [15] provides an overview of lattice techniques and [1] provides detailed descriptions of many forms of the LLL reduction algorithm. In this paper, we use LLL as a black box algorithm to find a vector of short Euclidean norm in a lattice defined by the $\mathbb{Z}$ span of the rows of a matrix.

We can strengthen the basic forgery attack described above by supplementing it with a lattice reduction technique. We exploit the fact that we have considerable freedom when choosing the constrained coefficients of s and t and make the observation that all possible simple forgeries differ from a given one by a $2N$-dimensional vector from an easily defined lattice. In other words, the idea here is (1) to generate an initial (s'',t'') using the basic forgery attack, and then (2) to correct some of the initial signature's deviations using lattice reduction. This hybrid approach allows us to generate forgeries averaging about 56 deviations in a few minutes.

Let (s'',t'') be the initial signature obtained using the basic forgery attack. Since $t''=s''*h \pmod{q}$, the vector (s'',t'') is in the lattice generated by the rows of the following matrix [1]:

$$L_{CS}=\begin{bmatrix} I_{(N)} & M_h \\ 0 & qI_{(N)} \end{bmatrix},$$

where $I_{(N)}$ denotes the N-dimensional identity matrix. In the basic forgery attack, to describe it in a slightly different fashion than previously, we found an invertible $k\times k$ submatrix U of

[1] Coppersmith and Shamir introduced this lattice in their attack on NTRU [4]. Since that time, the inventors of NTRU have hypothesized that the security of NTRU and NSS is related to the apparently hard problem of finding short vectors in this lattice.

M_h and then reordered the rows and columns of L_{CS} to obtain

$$L_{CS,2} = \begin{bmatrix} I_{(N-k)} & 0 & R & S \\ 0 & I_{(k)} & T & U \\ 0 & 0 & qI_{(N-k)} & 0 \\ 0 & 0 & 0 & qI_{(k)} \end{bmatrix} ,$$

where the invertibility of U made it easy to set the first k (actually, first $N-k$) and last k columns to whatever values we desired, modulo q. So, without loss of generality, we assume that in our initial signature (s'', t''), the first k coefficients of s'' and last k coefficients of t'' are chosen to be non-deviating (understanding that since the rows and columns of L_{CS} were reordered, s'' and t'' have been relabeled).

The attacker now would like to find some way of correcting the $N-k$ deviating coefficients of s'' (resp. t'') without touching the k non-deviating coefficients of s'' (resp. t''). To this end, the attacker would like to find a set of *harmless* row vectors in the lattice generated by $L_{CS,2}$ that contain zeros in the first k and last k positions, so that, for any vector (v_s, v_t) in this set, the pair $(s'' + v_s, t'' + v_t)$ will still be non-deviating in its first k and last k coefficients, while possibly having fewer deviations in its other positions.

We obtain the set of harmless vectors by making a slight modification to $L_{CS,2}$, obtaining a different lattice basis for the same lattice:

$$L_{CS,3} = \begin{bmatrix} I_{(N-k)} & -V & R-VT & 0 \\ 0 & I_{(k)} & T & U \\ 0 & qI_{(k)} & 0 & 0 \\ 0 & 0 & qI_{(N-k)} & 0 \\ 0 & 0 & 0 & qI_{(k)} \end{bmatrix} ,$$

where $V = SU^{-1} \pmod q$. To check that both generated lattices are indeed the same, one simply considers a linear combination of the first N rows of $L_{CS,2}$, corresponding to the sequence of coefficients $(\alpha_1, \cdots, \alpha_N)$. Writing the coefficients blockwise as (A_1, A_2), we see that exactly the same vector modulo q is obtained from the rows of $L_{CS,3}$ by a linear combination corresponding to $(A_1, A_2 + VA_1)$. The result follows. Notice that the rows $k+1$ to $N-k$ and rows $N+1$ to $2N$ of $L_{CS,3}$ have no nonzero coefficients in the first k or last k positions. We let $L_{harmless}$ be the lattice generated by these $(2N-2k)$ harmless vectors. These vectors are clearly linearly independent, so we conclude that the dimension of $L_{harmless}$ is exactly $(2N-2k)$.

Now, how do we use the lattice of harmless vectors to improve upon (s'', t'')? We will construct a lattice in which short vectors correspond to vectors with small deviations. Then we can search for a harmless vector, which, when added to (s'', t'') is a very short vector. This problem is an example of a closest vector lattice problem (CVP), related to the shortest vector lattice problem (SVP). See [15] for some comments on the relationship of the CVP to the SVP. To this end, we consider the lattice

$$L_{pq} = \begin{bmatrix} pL_{harmless} \\ (s', t') \end{bmatrix} ,$$

where (s', t') is the row vector with coefficients modulo pq satisfying $s' \equiv s'' \bmod q$ and $t' \equiv t'' \bmod q$, as well as $s' \equiv (f_0 * m) \bmod p$ and $t' \equiv (g_0 * m) \bmod p$ (again, and hereafter, keeping the relabeling in mind). For any row vector (v_s, v_t) in this lattice, $v_s * h = v_t \pmod q$. Moreover, v_s and v_t will satisfy one of three equations modulo p, depending on the value of the scalar coefficient of (s', t'):

$$v_s \equiv v_t \equiv 0 \bmod p, \text{ or}$$

$$v_s \equiv (f_0 * m) \bmod p \text{ and } v_t \equiv (g_0 * m) \bmod p, \text{ or}$$

$$-v_s \equiv (f_0 * m) \bmod p \text{ and } -v_t \equiv (g_0 * m) \bmod p \,.$$

If we could find a (v_s, v_t) with small coefficients — for example, in the range $(-q/2, q/2]$ — that does not satisfy the first condition[2] $v_s \equiv v_t \equiv 0 \bmod p$, then either (v_s, v_t) or $(-v_s, -v_t)$

[2] We observe in practice that this first condition may be avoided empirically with high probability via a small modifications of the lattice L_{pq}.

would be a valid forgery having zero deviations. Unfortunately, finding a short (v_s, v_t) appears to be a hard lattice problem that cannot be solved in any reasonable time for lattices as large as L_{pq}.

So, instead of attempting to reduce L_{pq}, we select c columns of L_{pq}, corresponding to unchosen coefficients of (s'', t''), and define L_{final} to be the submatrix of L_{pq} consisting of these c columns. The lattice generated by L_{final} is only c-dimensional. We then apply lattice reduction to L_{final}, obtaining a c-dimensional output vector. Every coefficient of the output vector that falls in the interval $(-q/2, q/2]$ is now non-deviating. In general, the expected number of deviations for s (resp. t) after this process is $(2N - 2k - c)/3 + \Delta/2$, where Δ is the expected number of coefficients of the c-dimensional output vector that are outside the interval $(-q/2, q/2]$.

For concreteness, when attacking the "practical implementation of NSS," the attacker might set k to be 95 and c to be 150 and reduce the resulting lattice using a blocksize of 20. The lattice reduction algorithm is completed a few minutes, and empirically, the resulting s and t typically each deviate in about 56 positions. For NSS to be secure, $D_{\max}$ would, of course, have to be set much lower than 56 to ensure that the hybrid forgery attack fails with high probability.

4 Transcript Attacks

4.1 Description of the Attack

In this section we show how to recover the private keys f and g by examining a transcript of signatures. A transcript consists of some number of pairs (m, s) of messages with valid signatures created by the NSS signature algorithm. We also obtain t for each message via the relation $t = s * h \pmod q$. The basis of the attack is to examine the distributions of the s or t coefficients for a subset of messages m. By setting one coefficient of m to a fixed value, the distributions of the coefficients of s and t converge to a limiting distribution which depends on a chosen coefficient of the secret key f or g. Thus we compare sample distributions of s or t to precomputed estimations of the limiting distribution for each possible value of f or g's coefficient.

As mentioned above, both the NTRU corporation research team and Mironov observed that if the *averages* of these distributions were dependent on the key coefficients, the private keys would be extremely rapidly recovered by essentially averaging the signatures. This problem was quickly corrected in the following version of NSS [8], by altering the signature algorithm to guarantee that the average of these distributions would be indeed independent of the private key coefficients. However certain s and t distributions do depend on the possible f and g coefficient values and are still quite distinct from one another. Comparing these distributions to one another or to a precomputed distribution leads to an exposure of the private key. One interpretation of the attack is that it is an exploitation of information leaked through the higher moments of the signatures.

The signature of a message s is obtained via an algorithm which chooses w_1 and w_2 according to an intricate algorithm (see [8]), and sets

$$s = f * (m + w_1 + pw_2).$$

This algorithm to choose w_1 and w_2 is described in Appendix B and it is easily observed to be constructed so as to avoid the simple averaging attack.

All of our experiments have used the suggested parameters $q = 128$, $p = 3$, and $N = 251$, although the technique is generally applicable. For this parameter set, the polynomials w_1 and w_2 have approximately 25 and 64 nonzero entries each, and m is set to have 32 coefficients equal to 1 and 32 equal to -1. The coefficients of s thus depend on the private key f, the message m and the randomly generated polynomials w_1 and w_2. The situation is entirely similar with g and t since

$$t = g * (m + w_1 + pw_2).$$

In order to obtain the coefficient f_k, we fix indices i_0 and j_0 with $i_0 = j_0 + k \bmod N$, and examine the distribution of s_{i_0} over a transcript of messages with m_{j_0}=1. Unraveling the convolution arithmetic, we have

$$s_{i_0} = \sum_{j+k=i_0} f_k(m_j + w_{1,j} + pw_{2,j}).$$

We note that the quantity $W_j = m_j + w_{1,j} + pw_{2,j}$ is nearly (but not exactly, due to a quirk of the w_1 generation) identically distributed for each index j, when the distribution is taken over random values of m. We consider s_{i_0} to be the sum of the random variables W_j, and because f has exactly 140 nonzero entries, s_{i_0} is nearly a sum of 140 identically distributed random variables drawn from a fixed distribution. However, requiring that $m_{j_0} = 1$ (or 0 or -1) distinguishes the random variable W_{j_0} from the others. Our observation is that the term $f_k W_{j_0}$ in the sum defining s_{i_0} will contribute differently depending on the value of f_k.

Since an explicit calculation of the distribution of s_i would necessarily rely on the complex formulas for w_1 and w_2, we tested the heuristic reasoning above with several numerical experiments. There are many possible variants of this approach. For example, one could also set $m_j = 0$ or $m_j = -1$ for the appropriate coefficient, and thereby extract additional information from a given size transcript. We mention here only one key optimization. Although we fixed the index i_0 above, in fact *every* coefficient of m may be potentially used to obtain information about each coefficient of f. Namely, for a single message-signature pair, examining s_i for all indices i such that $m_j = 1$ and $i + j = k$ speeds up the convergence by a factor of 32, since m has 32 coefficients equal to 1. Thus we essentially examine the distribution

$$s_{i,j} = \sum_{j+k=i, m_j=1} f_k(m_j + w_{1,j} + pw_{2,j})$$

over a large set of transcripts. We performed several computer experiments which implemented the above optimized statistical analysis. Our programs, written in C, were able to recover the private key with a very high degree of accuracy.

4.2 Efficiency of the Attack

To create the estimated background limiting distribution, we simply created several million messages, each signed by a different private key, and calculated the distributions of s_k conditional on $m_j = 1$, and f_k assuming a particular value in the set $\{-3, 0, 3\}$. These statistics were gathered individually for each coefficient of f_k, but for simplicity of exposition we combine them, and define the three probability distributions $\mathbf{F_0}$, $\mathbf{F_3}$, $\mathbf{F_{-3}}$, to be the limiting distributions of s_i given $m_j = 1$, and the prescribed f_k value.

Given a valid transcript of signed messages, for each coefficient index i, the sample distribution of s_i is formed, and denoted $\mathbf{S_i}$. Next $\mathbf{S_i}$ is compared to each of the distributions $\mathbf{F_0}$, $\mathbf{F_3}$, $\mathbf{F_{-3}}$, according to some distribution comparison method. To do this, we define $S_i(x)$ be the probability that $s_i = x$ for some $x \bmod q$. Similarly define $F_{0,i}(x)$, $F_{3,i}(x)$, and $F_{-3,i}(x)$ to be the respective probabilities that $s_i = x$ (conditional on the prescribed value of f_k and $m_j = 1$ for $i = j + k$). One simple, effective measure useful for distinguishing these distributions is defined as

$$\Delta_i(v) = \sum_x (F_{v,i}(x) - A_i(x))(S_i(x) - A_i(x)),$$

where $A_i(x)$ is the average of the frequencies ($F_{0,i}(x)$, $F_{3,i}(x)$, and $F_{-3,i}(x)$). Thus for each coefficient i of f, we calculate $\Delta_i(v)$ for $v \epsilon \{-3, 0, 3\}$. Next, we ordered the values $\Delta_i(3)$ and $\Delta_i(-3)$ and select the smallest 70 values to identify the coefficients with $f = 3$, and $f = -3$ respectively.

There are clearly many other ways in which the distributions could be compared, for example with the L_2 norm. The convergence obtained with the above metric efficiently recovered the key coefficients, and alternative measures were only used in subsequent confirming experiments. We briefly note that the first coefficient of f has a slightly different distribution than the other indices, but this may be easily adjusted for, and is of minimal importance as it is just a single index.

After predicting the private key, we compared it to the actual private key, and checked our results. Here we summarize the number of mistakes made for several applications of this technique to transcripts of different lengths.

Signatures	Trials	Average Errors
100,000	31	7.3
300,000	16	2.6
400,000	5	1.2

The incorrectly predicted coefficients all correspond to indices which were near the end of the 70 minimal values in the orderings of $\Delta_i(3)$ and $\Delta_i(-3)$. In fact, in each trial, we identified a subset of 40 such 'dubious' indices before comparing to the private key, and verified that all of the errors were located at such indices. Given this localization of the errors, we conclude that it is feasible via direct search to obtain the *exact* private key given our estimated private key.

Depending upon the size of the index subset to examine, we estimate that it is possible to obtain the exact key via direct search, even if the guess has up to 10 errors [3] Thus with our method of examining the s distribution, the key f may be completely deduced with as little as 100,000 signatures. We note also that significant partial information about a key's values may be used to greatly speed up certain lattice attacks, and in particular lattice reduction techniques may also be used to correct the estimated keys with a larger error tolerance than the brute force search method described above. These optimization techniques are not described further in this paper.

We note that it is likely that examining t rather than s would yield improved convergence rates. This conjecture is based on the fact that g is defined to have 80 nonzero entries rather than 140. We did not test this hypothesis directly in the above situation, but rather in the subsequent statistical attack on an NSS variant which we now describe.

4.3 An NSS Variant

Although the NSS version published in [8] was the subject of our first analysis, several variants proposed for the recent EESS standard [2] use a different private key structure. These key structures were proposed to increase the signing efficiency. Recall that the key space notation $\mathcal{L}(d,d)$ indicates a polynomial with d coefficients equal to 1 and d coefficients equal to -1. In the original version f was chosen to be $f = 1+3f_1$ where $f_1 \in \mathcal{L}(70,70)$, and $g = 1-2x+3g_1$ where $g_1 \in \mathcal{L}(40,40)$.

The optimized key space is formed as follows. $f = 1 + 3f_1 * f_2$ and $g = 1 + 2x + 3g_1 * g_2$, where $f_1 \in \mathcal{L}(7,7)$, $f_2 \in \mathcal{L}(5,5)$, $g_1 \in \mathcal{L}(5,5)$, and $g_2 \in \mathcal{L}(4,4)$.

Because of cancelation or correlation in the product, f and g typically contain fewer nonzero elements and contain several coefficients equal to 6 or -6. Thus while the original scheme has private keys with a known number of coefficients that assume values in the set $\{3,0,-3\}$, the new key have differing numbers of coefficients which typically assume values in the set $\{6,3,0,-3,-6\}$. (We ignore the first few indices of f and g for simplicity).

At first glance this appears to make the creation of the precomputed limiting distributions difficult. However, there are actually very few possible cases to consider. For example, a typical g has 62 coefficients equal to 3 or -3 and 5 equal to 6 or -6. The various other possibilities may be tried sequentially, in order of probability. Alternatively, we note that it is also true that the limiting distributions of s and t distinguish between the key structures with fewer or greater numbers of 6 and -6 coefficients very rapidly, without a need to fix values of m_j.

We found that the new private key structures led to even faster convergence. Several factors were changed simultaneously in the following experiment. First, we analyzed the distribution

[3] Assuming the 10 errors are so localized, an upper bound on the number of potential corrections to f is equal to the binomial coefficient $(40, 10)$, or less than 2^{29}.

of t instead of that of s. Secondly, we assumed the number of coefficients in $6, -6$ and $3, -3$ was known, and did not attempt to deduce it. Thirdly, we used the L_2 norm to compare the distributions. Finally, a two-stage algorithm first found the 6 and -6 coefficients (very easily), and the remaining indices were ordered by the L_2 distances to the precomputed distributions. The values of f_k were predicted according to this order. We found few errors in these predictions, with a smaller number of signatures.

Signatures	Trials	Average Errors
30,000	10	5.6
50,000	10	4.8
100,000	5	1.8
200,000	5	1.0

As with the standard keys, it is possible to identify a subset of questionable indices for which the guess may be in error. Therefore even a direct search is feasible to obtain the exact private key. Thus we conclude that this last technique would find the exact private key with a transcript of size 30,000.

Further optimizations are possible. For example, for a hybrid attack one may estimate both keys f and g via a method described above, and then assign confidence measures to each index. We then assume that the $N/2$ coefficients of f and $N/2$ coefficients of g that have the highest confidence measures are in fact correctly chosen. The remaining coefficients are determined by the relation $g = f * h$ as in section 3, and finally we check that the deduced key pair (f, g) is correct. Only enough signatures needed to provide half of each of f and g would be needed to obtain the exact key. Another promising optimization would be to use the value of the message coefficients m_j to make an educated guess to the values of m_j before they were reduced modulo q, and compare these distributions. Refinements of this strategy might reduce the number of signatures to ten or twenty thousand. However, in light of our efficient forgery and the fact that the NSS scheme has recently been replaced with a revised version, such optimizations are not pursued further in this paper.

5 Countermeasures

Subsequent to the discovery of these attacks, the authors of NSS began searching for a secure revision of the NSS signature scheme. Jeffrey Hoffstein outlined several techniques to alter the scheme at Eurocrypt 2001. These modifications were formalized shortly thereafter in a technical note on the NTRU web site [11], with further improvements in the second draft [12].

Shortly after this paper was initially submitted, the authors of NSS settled on a revision of NSS, complete with suggested parameter choices. The precise definition of the revised scheme may be found in a preliminary standards document [2]. Currently, the third draft of this standard is available at the Consortium for Efficient Embedded Security web page [3].

The revised scheme does indeed appear to resist the attacks described in this paper. We do not rigorously define the new scheme here, but only mention the revised scheme's salient features and how they obviate the above attacks. Further details may currently be found in technical notes, a preprint, and a standards document [11, 12, 10, 3].

The following is a partial list of the modifications.

1. **Private Key Generation:** In the version of NSS attacked in this paper, $f = f_0 + pf_1$ and $g = g_0 + pg_1$ where f_0 and g_0 are public parameters. In the revised scheme, $f = u + pf_1$ and $g = u + pg_1$ where u is kept private.
2. **Verification Criteria:** Verification is no longer based on the single criterion of deviations, but on multiple tests.
   – Norm Conditions: Verify that $|p^{-1}(s - m) \bmod q| < B$ and $|p^{-1}(t - m) \bmod q| < B$, where B is some bound on the *centered norms* [11].
   – Coefficient Distribution Checks: Perform a battery of specific checks (in [3]) on the distributions of the coefficients of s and t.
   – Moment Balancing: Optionally, use an alternate method of w_1 and w_2 creation, which alters the coefficients to include higher moment balancing.

These alterations were made to avoid the attacks presented in this paper, and therefore seem rather *ad hoc*. In particular, the verification protocol is strikingly lengthy [3], consisting of 17 steps! However the new key component u, norm conditions, and distributional criteria do appear to improve the security.

First, we discuss the new key component u. This is a very clever method of masking the combination of m coefficients which determine the distribution of w_0 values. Without the tool of controlling this distribution via selecting subsets of the messages, (say with $m_j = 1$) our transcript analysis can not effectively directly obtain distributions which are sensitive to the private key coefficient values. Adding u appears to make the distributions very close, even given millions of signatures. This renders the key recovery attack much less effective. Alternatively, the moment balancing techniques may also be used to make the distributions very close to one another.

Although the new verification protocol is a much less elegant revision than the use of u, it appears to serve its purpose of making forgery more difficult. The norm conditions relate the forgery problem of revised NSS to a (presumably hard) closest vector problem; the deviations criterion did not accomplish this. Also, the distribution checks appear to screen out forgeries generated by the forgery attacks above. However, it is unclear whether these new verification criteria are sufficient. It is likely that an attacker could already satisfy the norm conditions by simply using our (unmodified) forgery attack with the lattice reduction. Further cryptanalysis may show that it is possible to refine our attack to satisfy the distribution checks, as well.

The authors of NSS give some interesting analysis on how well the new scheme resists the attacks presented here [10]. They include a description of the new verification checks, a careful distributional analysis of the coefficients of the signatures in the new scheme, and a heuristic argument that signature forgery is as hard as a closest vector problem, assuming the adversary is given no transcript of previous signatures.

The new scheme is expected to receive renewed scrutiny, and since the key generation, signing and verification processes differ substantially, both forgery and key recovery techniques should be re-evaluated.

6 Conclusion

We wish to mention that our attack does not endanger the NTRU encryption scheme. On the other hand, we think that it shows the benefits of the *provable security* approach taken by cryptographic research in the last few years. NSS had no security proof at all, not even relative to a precisely described lattice problem of some form. Lacking such proof, one could not easily argue that NSS was immune to potential simple attacks, as demonstrated by the present work. Following the attack, NTRU researchers have investigated enhanced encoding/verification methods in [11]. It appears that such methods can offer a form of provable security by reducing forgery to solving a well defined lattice attack. This rules out the method of section 3. However, such a reduction would not apply to an attacker who takes advantage of transcripts of previously obtained signatures, as in section 4. We believe that the heuristic approach taken by NSS designers makes it extremely difficult to prevent such transcript attacks.

7 Acknowledgments

The authors would like to thank Julien P. Stern for help with a C-program and discussions, Philip Hirschhorn for providing real signature transcripts, Burt Kaliski, Phong Nguyen and Yiqun Lisa Yin for helpful discussions, and lastly Jeffrey Hoffstein, Jill Pipher, and Joseph Silverman who, after their conception of NSS, were also supportive of cryptanalysis research efforts.

References

1. H. Cohen. A Course in Computational Algebraic Number Theory. Graduate Texts in Mathematics, 138. Springer, 1993.

2. Consortium for Efficient Embedded Security. Efficient Embedded Security Standard (EESS) # 1: Draft 1.0. Previously posted on `http://www.ceesstandards.org`.
3. Consortium for Efficient Embedded Security. Efficient Embedded Security Standard (EESS) # 1: Draft 3.0. Available from `http://www.ceesstandards.org`.
4. D. Coppersmith and A. Shamir. Lattice Attacks on NTRU. In Proc. of Eurocrypt '97, LNCS 1233, pages 52–61. Springer-Verlag, 1997.
5. G. H. Hardy, E. M. Wright. An Introduction to the Theory of Numbers, 5th edition. Oxford University Press, 1979.
6. J. Hoffstein, J. Pipher and J.H. Silverman. NTRU: A New High Speed Public Key Cryptosystem. In Proc. of Algorithm Number Theory (ANTS III), LNCS 1423, pages 267–288. Springer-Verlag, 1998.
7. J. Hoffstein, J.H. Silverman. NSS: The NTRU Signature Scheme. Preliminary version, August 2000.
8. J. Hoffstein, J. Pipher, J.H. Silverman. NSS: The NTRU Signature Scheme. Preprint, November 2000. Available from `http://www.ntru.com`.
9. J. Hoffstein, J. Pipher, J.H. Silverman. NSS: The NTRU Signature Scheme. In Proc. of Eurocrypt '01, LNCS 2045, pages 211–228. Springer-Verlag, 2001.
10. J. Hoffstein, J. Pipher, J.H. Silverman. NSS: The NTRU Signature Scheme: Theory and Practice. Preprint, 2001. Available from `http://www.ntru.com`.
11. J. Hoffstein, J. Pipher, J.H. Silverman. Enhanced encoding and verification methods for the NTRU signature scheme. Previously posted on `http://www.ntru.com/technology/tech.technical.htm`.
12. J. Hoffstein, J. Pipher, J.H. Silverman. Enhanced encoding and verification methods for the NTRU signature scheme (ver. 2). May 30, 2001. Available from `http://www.ntru.com/technology/tech.technical.htm`.
13. A. Lenstra, H. Lenstra, and L. Lovasz. Factoring polynomials with rational coefficients. Math. Ann. 261, pages 515–534, 1982.
14. I. Mironov. A Note on Cryptanalysis of the Preliminary Version of the NTRU Signature Scheme. Preprint, January 2001. Available at `http://eprint.iacr.org/2001/005/`.
15. P. Nguyen and J. Stern. Lattice Reduction in Cryptology: An Update. In Proc. of Algorithm Number Theory (ANTS IV), LNCS 1838, pages 85–112. Springer-Verlag, 2000.
16. J. Stern. A method for finding codewords of small weight. Coding Theory and applications, LNCS 388, pages 106–113. Springer-Verlag, 1989.
17. R. Scheidler and H. C. Williams. A public-key cryptosystem utilizing cyclotomic fields. Designs, Codes and Cryptography 6, pages 117–131, 1995.

A An Example of Signature Forgery

Here we give an example of how to forge signatures using the public key. Let parameters be as defined in NSS251-3-SHA1-1 [2]; $N = 251$, $p = 3$, $q = 128$, $V_m = 32$, $\mathrm{Dev}^i_{\min} = 55$, $\mathrm{Dev}^i_{\max} = 87$, $f_0 = 1$, $g_0 = 1 - 2X$. Let the public key $h = f^{-1} * g \pmod q$ be

```
        0   1   2   3   4   5   6   7   8   9   a   b   c   d   e   f
--------------------------------------------------------------------
0 :     1  21 -59 -54   1 -33 -13 -11 -21  11 -30  31  -7  18 -61  85
1 :     3  41  52 -39 -30   4 -36  41 -11  56  46  -7  -7   7  -8  16
2 :   -58  -5  32  -3 -29  59  54 -25  53  48  47  32  -5  28  -9  -9
3 :    37  24 -50  17 -26 -58  10  39   4 -23 -55 -63 -29 -19   0  31
4 :    10  16 -25  28  29 -62  24  27  57  31  62 -61  35  39 -27   5
5 :    17 -22  22  28  32  41  14 -62 -18 -58  15  61  25   9  63  -9
6 :    47  30   0  58  58 -60  13  55   4   9 -62  11  58 -34 -39  13
7 :    40  27  36 -15  24 -31  37  23  31  55 -12 -20  43 -61   1  27
8 :   -44 -10  11  58 -63 -51 -46 -21  -6 -28 -17 -58 -28   6  21 -58
9 :    58  -3  10  -8 -26  48  12  64   2  14 -55 -20 -33 -24 -40   6
a :   -13  42  56 -23 -63  26 -52 -29  -4  35  12 -19 -24  47 -21  60
b :   -15 -17  63  62  55  17 -61   5  30  24 -32 -44  17  29 -63  57
c :    60 -25 -47 -51   2  11 -35 -44 -15  -5   7  -9  43  36 -18 -60
d :   -53  -2 -44  33 -27 -35 -17   5 -17  14   0   2  -6  49  29 -48
e :   -31  64  -8  64 -46  12  36 -57  23  -9  39  45  19  54 -21  49
f :    -7 -43  40  60 -45  20 -50   5  54 -13 -45
```

Let the message to be signed be

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0:	0	0	0	0	0	0	0	0	1	0	0	0	-	0	0	1	0	0	0	0	0	0	0	-	1	1	0	0	0	0	-	-
2:	0	0	0	0	0	0	0	-	0	0	0	1	0	1	0	0	0	0	0	0	-	0	0	0	0	1	0	-	-	0	0	-
4:	0	0	0	-	-	-	0	0	-	0	0	0	0	1	0	0	0	0	-	-	1	0	1	0	0	0	0	0	1	0	0	0
6:	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	-	1	0	0	0	0	-	0	0	0	1	0	0	-	0	0	0
8:	0	0	1	0	-	0	0	-	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	-	0
a:	0	0	1	-	-	0	0	0	0	0	0	-	-	-	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0	-	0	0
c:	0	0	0	0	0	-	0	0	0	0	0	0	0	0	1	1	0	1	-	0	-	0	0	0	-	0	0	1	0	0	0	0
e:	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-	0	0	0	1	0	0	0	0	0	1					

($-$ denotes the integer -1). We now find an initial signature $(s", t")$ by imposing $k = 95$ constraints on both s and t. For clarity in this example, we impose these constraints on the first 95 coefficients of $s"$ and last 95 coefficients of $t"$. Then, from the many possible $(s", t")$, we may get $s"$ equal to

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0 :	0	0	0	0	0	0	0	0	1	0	0	0	-1	0	0	1
1 :	0	0	0	0	0	0	0	-1	1	1	0	0	0	0	-1	-1
2 :	0	0	0	0	0	0	0	-1	0	0	0	1	0	1	0	0
3 :	0	0	0	0	-1	0	0	0	0	1	0	-1	-1	0	0	-1
4 :	0	0	0	-1	-1	-1	0	0	-1	0	0	0	0	1	0	0
5 :	0	0	-1	-1	1	0	1	0	0	0	0	0	1	0	56	0
6 :	-2	38	32	-41	-32	38	-4	-21	-4	8	-47	-57	-40	27	3	39
7 :	-44	14	33	52	-5	34	57	4	16	-4	-45	-18	-23	-58	-22	6
8 :	56	59	5	-57	-33	-55	19	-41	52	26	50	-54	2	57	-27	-30
9 :	47	9	36	-42	-17	-50	-7	-44	-55	-47	-30	-45	-39	34	36	7
a :	-32	-19	4	23	-43	-40	-3	59	22	-52	46	42	24	-12	-19	7
b :	24	-43	64	-41	54	-31	-13	-31	-49	-55	57	-54	-56	-60	-48	-20
c :	-36	26	4	18	16	-61	33	45	-16	53	59	64	-60	-13	35	-47
d :	-23	50	45	44	-52	53	49	-29	-52	35	54	53	-15	50	-18	26
e :	-7	-1	30	-50	-17	-14	-54	31	-59	35	-21	-44	-14	62	-15	-5
f :	36	27	-6	6	36	29	-12	1	58	19	21					

and $t"$ equal to

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0 :	25	-30	15	62	49	-24	-24	-12	15	-17	33	24	-61	64	-16	-57
1 :	-31	18	23	-29	27	39	-20	-35	-13	2	-54	39	36	-33	16	-13
2 :	-20	-45	-20	-3	25	10	54	-37	-33	41	-41	-47	-31	-15	31	-14
3 :	-52	16	-45	-10	-56	-22	-42	52	8	-20	55	13	30	32	-28	41
4 :	-57	25	49	-14	52	-38	-41	-35	22	-36	-27	-13	36	35	45	-10
5 :	54	-31	-9	3	-57	-37	9	-9	-16	-60	-59	14	18	26	-45	25
6 :	12	-40	11	31	41	5	-37	9	12	-21	-45	4	42	-18	-2	-29
7 :	-52	4	19	54	57	52	-23	-34	-31	-63	-60	-51	-14	42	2	13
8 :	56	-16	30	44	14	-37	-8	51	33	26	9	-12	-62	47	14	3
9 :	-50	18	-10	-33	24	-48	-4	60	-50	26	60	26	0	0	-1	-1
a :	0	0	1	0	1	-1	0	0	0	0	0	-1	1	1	-1	0
b :	0	1	1	0	0	1	1	0	0	0	1	-1	1	-1	-1	0
c :	0	0	0	0	0	-1	-1	0	0	0	0	0	0	0	1	-1
d :	1	1	0	-1	-1	-1	0	0	-1	-1	0	1	1	0	0	0
e :	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1
f :	0	-1	0	0	1	1	0	0	0	0	1					

The pattern of deviations between $s"$ and $(f_0 * m)$ looks as follows (each star denotes a deviation):

```
................................................................
```

```
..............................*.*******.**..*.**.*.***.***..***.
***.*******.*...**..******...*****...*.*********.***.*********.*
.**.*...****.**.****.***.*.*.*.***.***.***.***..*...**.****
```

For t" and $(g_0 * m)$ the pattern is:

```
...**...*....****.**..***.*...***.*.**..***.**.***.**..**..**.*.
****..****.*.***.***.****.****.*.******.***....**.*..******.****
****..****..***.***...*.**.*....................................
...........................................................
```

At this point s" and t" have 108 and 98 deviations, respectively. We now apply lattice reduction to coefficient positions 95 through 169 in s" and 81 through 155 in t" (the 75 leftmost coefficients in s" and 75 rightmost coefficients in t" that have not yet been constrained, for a total of 150 columns). For s, we get:

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0 :	0	0	0	0	0	0	0	0	4	0	0	0	-4	0	0	4
1 :	0	0	0	0	0	0	0	-4	4	4	0	0	0	0	-4	-4
2 :	0	0	0	0	0	0	0	-4	0	0	0	4	0	4	0	0
3 :	0	0	0	0	-4	0	0	0	0	4	0	-4	-4	0	0	-4
4 :	0	0	0	-4	-4	-4	0	0	-4	0	0	0	0	4	0	0
5 :	0	0	-4	-4	4	0	4	0	0	0	0	0	4	0	-30	0
6 :	9	-26	21	-54	39	33	21	-6	-47	9	43	-42	12	33	-50	-49
7 :	13	-24	3	6	-63	-19	12	33	6	7	-30	-36	-28	-12	-12	0
8 :	9	57	28	24	-52	12	18	20	6	-33	15	51	9	-33	-3	-9
9 :	-30	-17	-27	-21	6	9	-27	0	-36	-18	-42	-9	57	3	38	36
a :	36	-18	-47	35	47	-15	27	63	3	-12	-30	-22	-56	-40	10	57
b :	-49	-16	58	20	-53	-26	-19	29	-46	51	0	-49	-36	-29	15	-47
c :	-42	12	-52	51	40	47	42	-30	51	-53	-11	6	-39	51	13	-9
d :	-40	-51	-29	26	-32	44	3	27	-35	-9	55	-58	-60	0	-62	-17
e :	51	-26	30	-43	-50	7	-10	-8	-29	5	36	18	-30	-46	-21	42
f :	61	25	39	56	5	27	56	29	51	19	59					

For t we get:

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0 :	-37	41	-52	21	57	-57	-56	-63	-53	-2	5	40	-38	57	-62	18
1 :	7	57	-61	-32	18	-6	32	33	37	-30	36	62	-27	-15	54	-4
2 :	-34	-31	-51	24	-25	-8	62	57	38	28	-1	25	-50	-63	-63	-12
3 :	18	-10	-6	2	-39	-29	54	-13	-62	55	34	-35	-28	-60	-26	39
4 :	-2	-17	-44	-53	-38	-63	1	-19	-54	52	53	-61	50	10	-36	33
5 :	-27	-21	53	19	3	40	25	31	-33	-12	-54	-27	58	4	36	-15
6 :	21	-20	-5	-48	36	21	-30	42	4	-5	16	-56	0	-33	-41	-21
7 :	-39	1	30	18	-6	11	-43	6	-27	64	4	-6	-10	2	59	-3
8 :	30	-6	-8	-20	-31	20	3	17	-43	-15	-6	-15	15	-9	30	-3
9 :	-36	52	19	-3	-12	-9	-48	-48	27	-18	12	15	0	0	-4	-4
a :	0	0	4	0	4	-4	0	0	0	0	0	-4	4	4	-4	0
b :	0	4	4	0	0	4	4	0	0	0	4	-4	4	-4	-4	0
c :	0	0	0	0	0	-4	-4	0	0	0	0	0	0	0	4	-4
d :	4	4	0	-4	-4	-4	0	0	-4	-4	0	4	4	0	0	0
e :	0	4	4	0	0	0	0	0	0	0	0	0	0	0	0	4
f :	0	-4	0	0	4	4	0	0	0	0	4					

The deviation pattern for s is:

```
................................................................
................................................................
............................................*.*.*****.***.**.*.*
..*.*....**....*******..*.**..**...*******...*.***.**.**.**
```

The deviation pattern for t is:

```
***...*...******..**..******..**.*..****.**..**..*.***.**..*****
****.********.*...............................................*.
................................................................
...........................................................
```

Thus, we have produced an s and t that have 47 and 54 deviations from $(f_0 * m)$ and $(g_0 * m)$ respectively. These values are indeed even below the suggested parameter value of $\mathrm{Dev}^i_{\mathrm{min}} = 55$, which shows that our forgeries would pass even stricter deviation requirements.

Obviously the s and t of this example have highly unusual coefficient distributions modulo q, which the verifier could easily detect, but this need not be the case in general. We can make the coefficient distribution of s and t more ordinary by 1) constraining random coefficient positions and 2) distributing the values of the constrained coefficients of s" and t" more randomly modulo q, rather than setting them all equal to -1, 0 or 1.

B Determination of w_1 and w_2

The following pseudocode may also be found the appendix of [8]

```
let w2 have 32 +1's and 32 -1's
set w1[] to 0

compute s = f * (mes + 3 w2)
compute t = g * (mes + 3 w2)

reduce s and t modulo q
reduce s and t modulo p

//create w1, first try

for(i=0;i<N;i++)
  if(s[i] != mes[i] AND  t[i] != mes[i] AND  s[i] == t[i])
    w1[i] = (mes[i] - s[i]) mod p

  if(s[i] != mes[i] AND  t[i] != mes[i] AND  s[i] != t[i])
    w1[i] = 1 or -1 with 50% probability
loop

//create w1, second try

for(i=0;i<N;i++)
  if(s[i] != mes[i] AND  t[i] == mes[i])
     w1[i] = (mes[i] - s[i]) mod p with 1/4 probability

  if(s[i] == mes[i] AND  t[i] != mes[i])
     w1[i] = (mes[i] - t[i]) mod p with 1/4 probability

  if(w1 has more than 25 nonzero coefficients)
     break out of the loop
loop

// modify w2 to prevent averaging attack

for(i=0;i<N;i++)
    with probability 1/p, w2[i] = w2[i] - (mes[i] + w1[i])

w = w1 + 3 w2
```