Cryptanalysis of the NTRU signature scheme (NSS)

Jacques Stern

École normale supérieure
Département d'Informatique

Abstract. In 1996, a new cryptosystem called NTRU was introduced, related to the hardness of finding short vectors in specific lattices. This year, a signature scheme NSS (the NTRU Signature Scheme), apparently related to the same hard problem was proposed. In this note, we show that the problem on which the NSS scheme relies is much easier than anticipated and we decribe an attack that allows to efficiently forge a signature on any message. Following the attack, NTRU researchers have investigated enhanced encoding/verification methods in [4].

1 The setting

At the CRYPTO'96 rump session, a paper decribing a new cryptosystem called NTRU was distributed. An attack was quickly found by Coppersmith and Shamir (see [1]), which led the authors to adopt larger parameters. The current version of NTRU, as published in [3], remains unbroken. The NSS signature scheme appears in the paper [2], accepted for the EUROCRYPT'01 conference. An earlier version had been distributed at CRYPTO 2000 but a serious weakness in the preliminary scheme had been found by a Stanford student I. Mironov ([5]).

The public key of NSS consists of a polynomial h of degree $N-1$, whose coefficients are integers modulo q, where q is typically 128 and N is typically 251. Of course, there is a "trapdoor" but the exact way the trapdoor is used is irrelevant for us. There are also several additional parameters, whose role will appear further on. A signature s is a non zero polynomial of degree $N-1$, with coefficients modulo q. It is related to the message m to be signed, which takes the form of another polynomial of the same format. There are two conditions:

1. the distance $\Delta(s, f_0 * m)$ of s and $f_0 * m$ is between d and D
2. let $t = h * s$, then, the distance $\Delta(t, g_0 * m)$ of s and $g_0 * m$ is between d and D

In the above, f_0 and g_0 are fixed polynomials, typically $f_0 = 1$ and $g_0 = 1 - 2X$. Given two polynomials, A and B of the usual format, $\Delta(A, B)$ 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. $(-p/2, p/2]$) and p is typically 3. Note that two random polynomials have distance close to 167, whereas, typically, $d = 55$ and $D = 87$. Finally $*$ denotes multiplication modulo q and $X^N - 1$.

2 The attack

Paper [2] claims that a signature essentially proves possession of the secret trapdoor. Further, it envisions several potential attacks and conludes that the security of the system, with the above parameters, is comparable to RSA with 1024 bit moduli. However, it improperly views the underlying problem as a lattice problem. This is more of an error correction problem and, as demonstrated in many papers (see e.g. [6]), such problems take much larger dimensions to become hard.

The attack is very simple, once the perspective has been changed, as just indicated. Given the message to sign, the attacker chooses $k < N/2$ coordinates i and sets $s_i = (f_0 * m)_i$ for these coordinates. This leaves $N-k$ "free" coordinates. Similarly, the attacker chooses another set of k coordinates and decides that $t_j = (g_0 * m)_j$ for each such coordinate j. The adversary is *unlucky* if the constraints that he has fixed are incompatible. Note that these constraints form a set of k **linear** equations modulo q over the $N-k$ free unknowns. The coefficients of the unknowns in the system form a $k \times (N-k)$ matrix M 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 $(N-k)$ independant 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 vector of $M \bmod 2$. This precisely mean that these column vectors span the entire space.

We now assume that the attacker is lucky. Let s be a polynomial satisfying all the constraints he has fixed. We try to estimate $\Delta(s, f_0 * m)$. The constraints show that k of these coefficients are the same modulo q. They remain identical when reduced modulo q and next modulo p. We make another heuristic assumption that the other coefficients $s_i \bmod q \bmod p$ are random and independant elements of $\{-1, 0, 1\}$. Based on this assumption, $\Delta(s, f_0 * m)$ is the sum of $N-k$ Bernouilli trials with probability $2/3$. This is $\frac{2(N-k)}{3}$ on average. The same arguments applies to $\Delta(t, g_0 * m)$.

To conclude, we set $k = 121$. The attacker is lucky with probability at least $1 - 2^{-9}$. Granted this, he obtains a set of 128^9 solutions, which is a lot. The average values of $\Delta(s, f_0 * m)$, $\Delta(t, g_0 * m)$ for such solutions is 86.6. Based on our heuristic assumptions, we see that both values will be (slightly) smaller than the average with probability $\geq 1/4$. Thus, one of them yields the requested forged signature. Considering the large number of possible candidates, it is even possible to aim at values that lie close to the middle of the interval (d, D).

3 Efficiency of the attack

From the previous section, we see that the attack 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. Thus, preprocessing allows to compute the inverse of an appropriate invertible 121×121 submatrix and to perform the forgery using multiplication by this inverse. This makes forgery almost as fast as regular signature generation.

A C-program confirms the above. We have made two experiments:

1. the first with a public key that we manufactured, corresponding to the parameters from [2]
2. the second with a public key coming from one challenge from the NTRU WEB page. This challenge is for the encryption scheme. Unfortunately, there is no challenge for the signature scheme but we simply wished to make 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.

As shown by the appendix, we found forgeries whose distance pairs are respectively $(75, 74)$ and $(79, 79)$, which is quite in the middle of the interval (d, D).

4 Conclusion

We wish to mention that our attack does not endanger the NTRU encryption scheme. Nevertheless, we think that it shows the benefits of the *provable security* approach taken by cryptographic research in the last twenty five 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 immmune to potential simple attacks, as demonstrated by the present work.

5 Acknowledgements

We wish to thank Julien P. Stern for discussions and help with the C-program.

References

1. D. Coppersmith & A. Shamir. *Lattice Attacks on NTRU*, In Eurocrypt '97, Lecture Notes in Computer Science 1233, W. Fumy ed, Springer-Verlag, Berlin, 1997, 52–61.
2. J. Hoffstein, J. Pipher & J. Silverman. *NSS: The NTRU Signature Scheme*, In Eurocrypt '01, to appear.
3. J. Hoffstein, J. Pipher & J. Silverman. *NTRU: A new high speed public key cryptosystem*, In Algorithmic nNumber Theory (ANTS III), Portland, OR, June 1998, Lecture Notes in Computer Science 1423, J.P. Buhler ed, Springer-Verlag, Berlin, 1998, 267–268.
4. J. Hoffstein, J. Pipher & J. Silverman. Enhanced encoding and verification method for the NTRU signature scheme, availbale at `http://www.ntru.com/technology/tech.technical.htm`).
5. I. Mironov. A Note on Cryptanalysis of the Preliminary Version of the NTRU Signature Scheme, Cryptology ePrint Archive 2001/005.
6. J. Stern. *A method for finding codewords of small weight*, Coding Theory and applications, Lecture Notes in Comp Sci 388 Springer-Verlag, Berlin, 1989, 106-113.

Appendix: experiments

5.1 First experiment

```
Params: N=251 Q=128 P=3
Results: [74 73]
int pkey[] =
{48,48,11,24,121,91,75,91,5,84,38,60,74,70,54,70,20,66,68,112,24,24,48,
113,98,72,15,27,20,3,2,69,52,13,93,45,104,40,8,109,124,47,42,70,117,96,
12,9,34,80,122,58,104,42,43,75,114,58,102,7,61,104,76,113,117,41,30,93,
81,39,74,77,86,116,20,75,84,32,84,118,113,78,48,89,120,91,36,107,21,10,
114,83,114,62,68,103,103,99,68,56,10,15,5,96,3,25,43,88,58,127,78,43,78,
127,4,70,90,41,49,112,51,35,67,38,97,7,13,72,106,82,0,116,97,6,84,100,31,
127,60,89,127,11,4,77,10,9,19,100,50,69,84,101,104,23,11,74,31,25,18,9,
107,19,126,76,25,82,48,56,82,109,18,81,120,22,30,2,31,49,102,81,118,59,
55,95,82,66,41,113,91,59,123,70,78,121,18,103,75,67,32,29,48,50,110,40,
72,12,42,104,62,16,57,52,75,112,19,30,51,60,15,14,120,10,85,70,3,103,46,
79,42,78,108,90,0,91,2,72,103,44,48,37,61,106,90,8,90,109,38,13,42,54,28}


int mess[] =
{0,0,0,0,127,0,127,0,1,0,0,127,127,0,0,0,0,127,0,1,0,0,0,0,0,0,127,0,
127,0,1,0,0,0,0,0,0,0,0,0,0,127,0,0,127,0,0,0,0,0,0,0,0,0,0,0,127,127,
0,1,0,0,0,0,1,0,0,1,0,127,0,0,0,0,0,127,0,0,0,0,0,0,0,0,1,0,0,0,127,0,0,
0,0,0,0,0,1,1,0,127,0,127,1,1,0,0,0,0,1,0,0,0,0,0,0,127,0,127,0,0,0,0,0,
0,1,1,0,127,0,0,0,0,127,1,0,0,0,127,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,
0,0,1,0,0,0,0,0,0,0,0,127,0,127,0,0,0,1,1,127,0,0,0,0,0,127,0,0,127,0,0,
0,1,0,0,127,127,0,0,0,0,0,127,1,0,1,1,0,127,0,0,0,1,1,0,1,0,0,0,0,127,1,
1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,127,0,0,0,0,0,0}

int sign[] =
{10,87,103,109,112,17,74,89,15,0,55,97,61,27,6,54,105,61,107,39,77,16,90,
8,33,87,22,77,27,1,109,10,98,114,74,61,69,86,13,36,63,45,69,68,51,75,64,
93,94,118,68,12,9,34,19,13,4,118,45,35,33,100,58,27,53,89,97,29,107,101,
98,44,60,109,89,117,76,35,97,110,6,98,126,88,8,22,83,2,116,33,121,98,23,
36,94,43,21,20,117,19,69,76,110,121,4,0,79,58,113,1,124,92,67,47,77,62,
28,18,72,2,80,0,0,127,1,1,127,0,127,127,92,30,48,107,53,96,71,95,3,109,
27,1,113,101,15,21,60,27,21,95,77,10,108,104,63,42,84,3,9,63,22,12,125,
9,60,39,39,18,77,14,113,79,42,0,107,40,66,50,57,104,42,30,74,35,119,24,
106,68,9,51,87,98,104,100,118,6,12,42,77,3,50,55,63,58,90,86,17,9,110,
116,55,31,27,111,18,110,6,119,26,123,90,15,119,107,48,81,122,48,48,42,
110,122,107,18,45,54,71,101,24,48,92,18,42,101,2,71,110,42,30,60,113}
```

5.2 Second experiment

```
Params: N=263,P=128,Q=3
Results: distance = [75 76]

int pkey[]=
{4,5,7,18,5,114,63,126,109,30,36,127,57,103,71,57,16,46,34,29,4,123,55,
110,106,70,94,33,67,89,7,107,85,59,49,91,72,13,62,27,53,66,79,57,5,25,93,
4,31,9,38,59,2,52,66,5,105,70,96,92,9,59,34,68,92,24,113,78,64,98,111,64,
80,116,8,94,74,26,53,21,69,88,123,83,101,0,82,126,54,122,26,66,79,29,89,
82,113,21,55,62,113,6,22,122,60,47,58,93,68,65,19,8,4,116,41,89,118,90,
103,12,84,9,112,51,57,70,76,81,35,47,68,93,24,34,7,68,13,30,48,74,93,36,
48,118,14,125,45,103,85,37,108,81,23,30,90,72,54,48,96,57,124,124,40,48,
9,11,10,10,99,100,106,81,84,52,47,114,93,100,34,44,21,118,116,55,81,97,
25,15,100,71,73,96,34,57,30,111,57,28,51,105,35,31,97,85,68,68,4,59,33,
107,45,41,25,10,2,1,63,54,29,100,13,45,71,102,44,82,93,78,19,61,121,123,
109,50,37,5,95,16,69,3,17,67,49,2,68,82,12,17,59,68,7,115,57,21,78,1,74,
80,105,104,63,21,0}

int mess[] =
{0,0,127,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,127,0,0,0,0,0,0,0,1,0,1,0,0,0,1,
0,1,0,0,0,0,0,0,1,0,127,0,0,0,0,127,0,0,0,1,0,0,0,0,0,0,0,1,127,127,127,0,
0,0,0,0,0,127,0,1,0,0,0,127,0,1,1,1,1,0,0,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,
127,0,127,0,0,0,0,127,0,0,0,127,0,127,0,0,1,1,0,0,0,0,0,0,0,0,0,127,0,0,
1,0,0,0,0,0,127,0,0,1,0,0,0,0,0,127,0,0,0,1,0,0,0,0,127,0,0,0,127,0,0,127,
0,127,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,127,0,0,0,0,127,1,0,0,0,127,0,127,
127,1,0,0,0,0,0,0,0,0,0,0,0,127,1,0,0,0,0,1,0,0,127,0,127,127,0,0,0,0,0,0,
0,0,0,127,0,0,1,0,0,0,0,0,1,0,0,0,127,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,
0,0}

int sign[] =
{127,2,47,24,71,92,78,121,25,100,96,114,115,79,24,54,105,100,118,23,40,
19,123,50,122,15,4,87,14,60,114,17,22,66,98,52,17,51,6,120,79,125,92,18,
52,85,63,104,9,97,34,70,30,96,87,42,104,17,29,9,118,35,71,100,9,61,65,54,
115,60,47,30,80,86,64,117,38,121,89,106,67,72,31,99,115,49,102,83,102,22,
33,65,92,60,54,8,42,108,51,16,4,117,113,60,53,83,19,64,20,43,29,92,34,9,
16,117,30,37,23,101,74,94,92,33,17,1,0,1,0,127,0,127,127,127,127,127,127,
127,109,27,80,126,18,3,113,24,89,106,48,48,104,31,24,60,27,18,70,80,36,9,
8,51,48,41,98,76,33,98,122,3,110,31,45,33,39,3,68,28,15,57,122,30,18,70,
65,92,0,92,94,58,12,119,45,2,80,88,121,7,83,98,27,48,65,63,51,125,98,71,
41,22,110,89,98,30,117,116,101,79,89,53,67,54,30,12,63,45,12,33,95,0,121,
68,110,69,15,74,119,33,119,25,71,15,107,121,42,113,0,86,92,51,12,74,74,
54,69,116,63,107,68,110,27,6,110}
```