# NSS: An NTRU Lattice-Based Signature Scheme

Jeffrey Hoffstein, Jill Pipher, and 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-based cryptography, NTRU, NSS

## 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 SmartCards 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 [1,2,6,7,9,11,12,15,16,17].

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.

In the original version of this paper for Eurocrypt 2001, we both introduced NSS and optimized it for maximum efficiency and minimum signature length. As a result the underlying ideas and security analysis were less transparent than they might have been. To alleviate this problem and attempt to address some of the concerns of the referees, the present paper takes the following form. We first present a complete version of NSS and a set of parameters optimized to provide security comparable to RSA 1024 along with high efficiency. We then describe the properties of an implementation of this system at these parameters. The version of this paper originally submitted to Eurocrypt then provided a security analysis tailored specifically to these parameters. In the current version

B. Pfitzmann (Ed.): EUROCRYPT 2001, LNCS 2045, pp. 211–228, 2001.

we eliminate some details of the security analysis of the optimized version in order to include a discussion of the less efficient version. In this way we hope to elucidate the main ideas underlying NSS and thereby make this paper easier to read. Complete details of the analysis of the optimized version are available on our website at `<www.ntru.com/technology/tech.technical.htm>`.

We also note that the signature scheme described in this paper differs in some respects from the scheme described by Jeff Hoffstein at the CRYPTO 2000 rump session. In order to optimize NSS, the rump session version used disparate sized coefficients whose existence was concealed by allowing $p$ to divide $q$, which led to a statistical weakness. (This weakness was independently noted by Mironov [10].) The use of uniform coefficients and relatively prime values for $p$ and $q$ makes NSS more closely resemble the original NTRU public key cryptosystem, a system that has withstood intense scrunity since its introduction at CRYPTO ′96.

The authors would like to thank Phil Hirschhorn 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 assume some familiarity with [4], but we repeat definitions and concepts when it appears 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

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

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

*Remark 1.* For ease of exposition we 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 NSS are formed as follows. Bob begins by choosing two polynomials $f$ and $g$ having the form

$$f = f_0 + pf_1 \qquad \text{and} \qquad g = g_0 + pg_1. \tag{2}$$

Here $f_0$ and $g_0$ are fixed universal polynomials (e.g., $f_0 = 1$ and $g_0 = 1 - 2X$) 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 polynomial $f$.

Before describing exactly how NSS works, we would like 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 $f$ (i.e., $f$ has coefficients that have small absolute value compared to $q$) with the property that the product $g \equiv f * h \pmod{q}$ also has small coefficients. Equivalently (see Section 4.2), 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 $f$ without actually revealing its value 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 following quantity.

**Definition 1.** *Let $a(X)$ and $b(X)$ be two polynomials in $R$. First reduce their coefficients modulo $q$ to lie between $-q/2$ to $q/2$, then reduce their coefficients modulo $p$ to lie in the range between $-p/2$ and $p/2$. If*

$$\bar{a}(X) = \bar{a}_0 + \cdots + \bar{a}_{N-1}X^{N-1} \quad \textit{and} \quad \bar{b}(X) = \bar{b}_0 + \cdots + \bar{b}_{N-1}X^{N-1}$$

*are the reductions of $a$ and $b$, respectively, then the* deviation of $a$ and $b$ *is*

$$\mathrm{Dev}(a, b) = \#\{i : \bar{a}_i \neq \bar{b}_i\}.$$

*Intuitively,* $\mathrm{Dev}(a, b)$ *is the number of coefficients of $a$* mod $q$ *and $b$* mod $q$ *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 $h \equiv f^{-1} * g \bmod q$ 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 4.9.) 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 describe later, see Section 2.1. 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 checks that $s \neq 0$ and then verifies the following two conditions:

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

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

(B) Alice uses Bob's public verification key $h$ to compute the polynomial $t \equiv h * s \pmod{q}$. She then checks if the deviation of $t$ from $g_0 * m$ satisfies

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

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

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 [5] 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 $N$.

This is the key difference between the optimized version of NSS presented in the next section and a somewhat less efficient version. If we take $D_{\min} = D_{\max} = 0$, i.e., if we allow no deviations, then a transcript will reveal $f * w$ and $g * w$ exactly. Lattices of dimension $N$ can be reduced faster than lattices of dimension $2N$. Consequently, for a secure version of NSS assuming no deviations we require a larger value of $N$. We will show that if $N$ is chosen greater than about 700 this still gives a fast and equally secure signature scheme, albeit with somewhat larger key and signature sizes than the optimized version of NSS described in this note.

This concludes our overview of how NSS works. In the next section we suggest a parameter set and explain why we believe that it provides a level of security comparable to RSA 1024. Table 1 compares the efficiency of NSS to other systems. In the following sections we provide a security analysis, although due to space constraints, we refer the reader to [5] for some details, especially for the optimized version with $D_{\min}, D_{\max} > 0$.

## 2 A Practical Implementation of NSS

The following parameter selection for NSS appears to create a scheme with a breaking time of at least $10^{12}$ MIPS years:

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

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

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

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 $0 \le e_i < 251$, and then $m = \sum_{i=1}^{32} X^{e_i} - \sum_{i=33}^{64} X^{e_i}$.

The polynomial $w$ has the form $w = m + w_1 + pw_2$, so 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 4.4) or gaining information via a reversal averaging attack (see Section 4.6). 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 $m + w_1$. For the parameter set (3) given above, the polynomial $w_1$ has up to 25 nonzero coefficients and $w_2$ is initially chosen at random from the set $\mathcal{T}(32)$. The precise prescription for creating $w$ is described in Section 2.1.

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.

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

| | 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 |

*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 [8] 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.

### 2.1 Selection of the Masking Polynomial $w$

The polynomial $w = m+w_1+pw_2$ has two purposes. First, it includes the message digest $m$ and is thus the means by which $m$ is attached to the signature $s$. Second, it contains polynomials $w_1$ and $w_2$ that introduce variability into the signature and prevent an attacker from gaining useful information that might be used to find the private key $f$ or to directly forge a signature.

There are two principle areas that must be addressed when selecting $w$. First, in the optimized version we must ensure that an attacker cannot lift the values of $s \equiv f * w \pmod{q}$ and $t \equiv g * w \pmod{q}$ to the exact values of $f * w$ or $g * w$ in $\mathbb{Z}[X]$. Second, we must ensure that the attacker cannot use averages formed from long transcripts of signatures to deduce information about $f$ or $g$.

The first item is addressed by selecting $w_1$ so as to alter many of the coefficients of $f * (m + pw_2)$ and $g * (m + pw_2)$ that lie outside the range from $-q/2$ to $q/2$. This has the effect of masking the coefficients that have suffered nontrivial reduction modulo $q$ and prevents the attacker from undoing the reduction. The second item is handled by changing $1/p$ of the coefficients of $w_2$; this has the effect of forcing all second moment transcript averages to converge to 0. We now describe exactly how $w_1$ and $w_2$ are created. For ease of exposition, we assume that $p = 3$. For further details of why this procedure protects against lifting and averaging attacks, see [5].

The first step is to choose a random polynomial $w_2 \in \mathcal{T}(d_{w_2})$. That is, $w_2$ has a specified number of 1's and $-1$'s. For example, the parameter set (3) takes $w_2 \in \mathcal{T}(32)$. The next step is to compute preliminary signature polynomials

$$s' \equiv f * (m + pw_2) \pmod{q} \qquad \text{and} \qquad t' \equiv g * (m + pw_2) \pmod{q}. \qquad (4)$$

Next we choose $w_1$. We start with $w_1 = 0$. We let $i = 0, 1, 2, \ldots, N-1$ and run through the coefficients $s'_i$ and $t'_i$ of $s'$ and $t'$, performing the following steps. [The quantity $\texttt{w}_1\texttt{-Limit}$ used below is a prespecified parameter. For the parameter set (3), its value is 25.]

- If $s'_i \not\equiv m_i \pmod{p}$ and $t'_i \not\equiv m_i \pmod{p}$ and $s'_i \equiv t'_i \pmod{p}$, then set $w_{1,i} \equiv m_i - s'_i \pmod{p}$.
- If $s'_i \not\equiv m_i \pmod{p}$ and $t'_i \not\equiv m_i \pmod{p}$ and $s'_i \not\equiv t'_i \pmod{p}$, then set $w_{1,i} = 1$ or $-1$ at random.
- If $s'_i \not\equiv m_i \pmod{p}$ and $t'_i \equiv m_i \pmod{p}$, then with probability 25%, set $w_{1,i} \equiv m_i - s'_i \pmod{p}$.
- If $s'_i \equiv m_i \pmod{p}$ and $t'_i \not\equiv m_i \pmod{p}$, then with probability 25%, set $w_{1,i} \equiv m_i - t'_i \pmod{p}$.
- If $i = N - 1$ or if $w_1(X)$ has more than $\mathtt{w_1}$`-Limit` nonzero coordinates, the construction of $w_1$ is complete.

Finally, we need to make some alterations to $w_2$ to prevent the averaging of long transcripts of signatures. This is done by taking each coefficient $w_{2,i}$, $0 \le i < N$, and with probability 1/3, replacing it with with $w_{2,i} - m_i - w_{1,i}$.

This completes the description of how $w_1$ and $w_2$ are chosen.

## 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).

### 3.1 The Norm of a Polynomial

In order to analyze the two verification conditions we briefly digress to discuss norms of polynomials.

Let

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

be a polynomial with integer coefficients and let $\mu$ be the average of the coefficients. We define the *centered Euclidean Norm* and the *Sup Norm* of $a$, denoted respectively $\|a\|$ and $\|a\|_\infty$, by the formulas

$$\|a\| = \sqrt{(a_0 - \mu)^2 + \cdots + (a_{N-1} - \mu)^2} \quad \text{and} \quad \|a\|_\infty = \max\bigl\{|a_0|, \ldots, |a_{N-1}|\bigr\}.$$

In our examples, $\mu$ will be close to or equal to zero.

We require certain facts about polynomials with small coefficients. For random polynomials with small coefficients such as $f$ and $w$, it is generally true that

$$\|f * w\| \approx \|f\| \cdot \|w\| \quad \text{and} \quad \|f * w\|_\infty \approx \gamma \|f\| \cdot \|w\|, \tag{5}$$

where $\gamma < 0.15$ for $N < 1000$. The NTRU cryptosystem relies on these properties of small polynomials, which are discussed in [4]. (Note that the infinity norm defined in [4] is actually twice the infinity norm defined here.)

With this background we now easily check the completeness of NSS.

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

$$\begin{aligned} s &\equiv f * w \pmod{q} \\ &\equiv (f_0 + pf_1)(m + w_1 + pw_2) \pmod{q} \\ &\equiv f_0 * m + f_0 * w_1 + pf_0 * w_2 + pf_1 * w \pmod{q}. \end{aligned}$$

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

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

The estimates in (5) tell us that before reduction modulo $q$, the absolute value of the coefficents of $f * w$ is bounded above by $\gamma \|f\| \cdot \|w\|$. As long as this quantity does not greatly exceed $q/2$, little reduction modulo $q$ will take place. 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 $m \bmod p$. Alternatively, if $\|f\|$ and $\|w\|$ are sufficiently small, then no reduction modulo $q$ will take place and one can set $D_{\min} = D_{\max} = 0$. Thus Bob's signature will pass 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. For the parameters $(N, p, q, D_{\min}, D_{\max}) = (251, 3, 128, 55, 87)$ from Section 2, we see from Table 2 that the probability that $\mathrm{Dev}(s, f_0 * m)$ is valid is approximately 87.33% and the probability that $\mathrm{Dev}(t, g_0 * m)$ is valid is approximately 90.92%. Thus 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 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

**Table 2.** Deviations Between $f_0 * m$ and $s$ and Between $g_0 * m$ and $t$

| Range | $\mathrm{Dev}(s, f_0 * m)$ | $\mathrm{Dev}(t, g_0 * m)$ |
|---|---:|---:|
| 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

of valid signatures. He might also try to forge a signature on a message without first finding the private key. We describe the hard lattice problems that underlie some of these attacks and examine the success probabilities of other attacks that rely on random searches. In all cases we 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 must refer the reader to [5] for many of the technical details related to the analysis of the optimized parameter set.

### 4.1 Random Search for a Valid Signature on a Given Message

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

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

If $D_{\min} = D_{\max} = 0$ these conditions become:

(A′) $s \equiv f_0 * m \pmod{p}$.
(B′) $t \equiv h * s \pmod{q}$ satisfies $t \equiv g_0 * m \pmod{p}$.

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

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

(Notice that for condition (B′), the probability is $p^{-N}$, since all $N$ "random" coefficients of $t \pmod{p}$ must match $g_0 * m$.) Table 3 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 $2^{-80.95}$.

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

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

## 4.2 NTRU Lattices and Lattice Attacks on the Public Key

Oscar can try to extract the private key $f$ from the public key $h$ with or without 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 discuss attempts by Oscar to obtain the private key from the public key by lattice reduction methods. As is the case with the NTRU cryptosystem, recovery of the private key by this means is equivalent to solving a certain class of shortest or closest vector problems.

We begin with a brief exposition of our approach to the analysis of lattice reduction problems. We have perfomed 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 [3,4,6,7].)

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

$$\kappa = \kappa(L, r) = \frac{r}{d^{1/n}\sqrt{n/(2\pi e)}}. \tag{6}$$

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 the gaussian heuristic says that a solution, if one exists at all, will probably be unique (or unique up to obvious symmetries of the lattice). The closer that $\kappa$ is to 0, the easier it will be to find the unique solution using lattice reduction methods. As $\kappa$ gets close to 1, lattice reduction methods become less effective.

For example, let $(L_n, r_n, v_{0,n})$ be a sequence of lattices, radii, and target vectors of increasing dimension $n$ that contain a target vector $v_n \in L_n$ (i.e., satisfying $|v_n - v_{0,n}| < r_n$) and whose $\kappa$ values satisfy

$$\kappa_n = \kappa(L_n, r_n) = c/\sqrt{n} \tag{7}$$

for a constant $c$. Then our experiments suggest that the time necessary for lattice reduction methods to find the target vector $v_n$ grows like $e^{\alpha n}$ for a value of $\alpha$ that is roughly proportional to $c$. Similarly, if $\kappa \geq 1$, then a solution will probably not be unique, but it becomes progressively harder to find a solution as $\kappa$ 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. The lattices associated to these parameter sets are then subjected to extensive experimental testing.

Recall from (2) that the public key 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 can be used to try to derive $f$ and $g$ from $h$. See [4,13] 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 $L^{\mathrm{NT}}$ consists of the linear combinations of the $2N$ vectors in the set

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

Equivalently, $L^{\mathrm{NT}}$ 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 $L^{\mathrm{NT}}$ and will be shorter than the expected shortest vector of $L^{\mathrm{NT}}$ (i.e., $\kappa < 1$). Thus in principle, $(f, g)$ should be essentially unique and findable by lattice reduction methods.

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 the same $2N$-dimensional lattice The object is to search for the vector in $L^{\mathrm{NT}}$ that is closest to the vector $(0, (g_0 - f_0 * h)p')$, where $pp' \equiv 1 \pmod{q}$. If successful, this attack produces a small $F$ such that

$G \equiv F * h - (g_0 - f_0 * h)p' \pmod{q}$ is also small. Then $(f_0 + pF, g_0 + pG)$ is either the original key or a useful substitute. With this approach, after balancing the lattice as in [4], we obtain the following estimate for the constant $c$ in equation (7):

$$c > 2\sqrt{\pi e \|f_1\| \|g_1\| / q}. \tag{8}$$

Experimental evidence shows that if $L$ runs through a sequence of NTRU type lattices of dimension $2N$ with $N > 80$ and $q \approx N/2$ and if the constant $c$ of (7) satisfies $c > 3.7$, then the time $T$ (in MIPS-years) necessary for the LLL reduction algorithm to find a useful solution to the closest vector problem satisfies

$$\log T \geq 0.1707N - 15.82. \tag{9}$$

Thus if $N = 251$ and $c = 3.7$, one has $T > 5 \cdot 10^{11}$ MIPS-years.

For the optimized version of NSS presented in Section 2, we have $N = 251$ and $c > 5.3$. Since larger $c$ values in (7) yield longer LLL running times, we see that the time to find the target vector should be at least $10^{12}$ MIPS-years, and is probably considerably higher. In general, we obtain this lower bound provided that $N, \mathcal{F}_f, \mathcal{F}_g$ are chosen so that $\|f_1\|, \|g_1\|$ give a large enough value for $c$ in (8).

### 4.3 Lattice Attacks on Transcripts

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

$$fw, fw', fw'', \ldots \bmod q \qquad \text{and} \qquad gw, gw', gw'', \ldots \bmod q. \tag{10}$$

If Oscar can determine any of the $w$ values, then he can easily recover $f$ and $g$. Using division, Oscar can obtain $w^{-1}w' \bmod q$ and other similar ratios, so he can launch an attack on the pair $(w, w')$ identical to that described in the preceding section. As long as $\|w\|$, $\|f\|$, and $\|g\|$ are about the same size, the value of $\kappa$ will remain the same or increase, leading to no improvement in the breaking time.

Oscar can also set up a $kN$-dimensional NTRU type lattice using the ratios of signatures $w^{(1)}/w^{(1)}, w^{(2)}/w^{(1)}, \ldots, w^{(k)}/w^{(1)}$. The target is $(w^{(1)}, \ldots, w^{(k)})$. With this approach the value of $\kappa$ decreases as $k$ increases, giving the attacker a potential advantage, but the increasing dimension more than offsets any advantage gained. With the parameters given in Section 2, the optimal value of $k$ for the attacker is $k = 10$, giving $\kappa = 4.87/\sqrt{10N}$. This is a bit better than the $c > 5.3$ coming from the original $2N$ dimensional lattice, but still considerably worse than the $c = 3.7$ that gave us the original lower bound of $10^{12}$ MIPS-years.

There are several other variations on the lattice attacks described in this and the previous section, but none appears to be stronger then the closest vector attack on the public key given in Section 4.2.

### 4.4 Lifting a NSS Signature Lattice to $\mathbb{Z}$

Recall that an attacker Oscar is presumed to have access to a transcript of signed messages such as given in (10). Various ways in which he might try to exploit this mod $q$ information are described in Sections 4.3. In this section we are concerned with the possibility that Oscar might lift the transcript information (10) and recover the values of $f * w, f * w', \dots$ exactly over $\mathbb{Z}$.

This is the primary area where the signature scheme with zero deviations differs from the optimized scheme. If the signatures can be recovered over $\mathbb{Z}$, as they can be if $D_{\min} = D_{\max} = 0$, then two additional lattice attacks are made possible. In the optimized scheme of Section 2, we ensure that a lift back to $\mathbb{Z}$ is impractical by making the number of possible liftings greater than $2^{80}$. This leaves Oscar with only the lattice attacks described in Sections 4.2 and 4.3 and allows us to take $N = 251$ while maintaing a breaking time in excess of $10^{12}$ MIPS years.

We now investigate the attacks that are possible if such a lifting can be accomplished. This analysis, irrelevant for the optimized parameters, allows us to set parameters for a simpler variant of NSS with $D_{\min} = D_{\max} = 0$.

Suppose that Oscar forms the lattice $L'$ generated by $X^i * f * w$ with $0 \le i < N$ and a few different values of $w$ (or similarly for $X^i * g * w$). It is highly likely that the shortest vectors in $L'$ are the rotations of $f$. Essentially, Oscar is searching for a greatest common divisor of the products $f * w$, though the exponentially large class number of the underlying cyclotomic field greatly obstructs the search. Although it is still not easy to find very short vectors in the lattice $L'$ using lattice reduction, the fact that $\dim(L') = N$, as compared to the NTRU lattice $L^{\mathrm{NT}}$ of dimension $2N$, means that $L'$ is easier to reduce than $L^{\mathrm{NT}}$.

The difficulty of finding a solution to the shortest vector problem for the lattice $L'$ appears to be related, as one might expect, to the magnitude of the norm of $f$. For example, if one considers a sequence of lattices $L'$ of dimension $N$ formed with $f$ satisfying $\|f\| \approx \sqrt{2N/3}$, then our experiments have shown that the extrapolated time necessary for the LLL reduction algorithm to locate $f$ is at least $T$ MIPS years, where $T$ is given by the formula

$$\log T = 0.1151N - 7.9530. \tag{11}$$

As the norm of $f$ is reduced, the time goes down. For example, if we take $\|f\| \approx \sqrt{0.068N}$, then our experiments show that the breaking time is greater than the $T$ given by the formula

$$\log T = 0.0785N - 6.2305. \tag{12}$$

One further lattice attack of dimension $2N$ is enabled if a lifting to $\mathbb{Z}$ is possible. One can view it as an alternative attack on the gcd problem. Given two products $f * w$ and $g * w$, one can reduce these modulo any integer $Q$ and then take the ratio, obtaining $f^{-1} * g$ modulo $Q$. This is very similar to the original problem of finding the private key from the public key, but there is an important difference. The integer $Q$ can be chosen as large as desired, which has the effect of decreasing the value of $\kappa$. As a result, it becomes easier to reduce the lattice. The advantage of making $Q$ larger does not continue indefinitely, and the ultimate result is to reduce the effective dimension of the lattice from $2N$ to $N$. Experiments have shown that when $f$ and $g$ satisfying $\|f\| = \|g\| = \sqrt{2N/3}$ are used to generate these lattices and an optimal value of $Q$ is chosen for each $N$, the extrapolated time necessary for the LLL reduction algorithm to locate $f$ is at least $T$ MIPS years, where $T$ is given by the formula

$$\log T = 0.0549N + 1.7693. \tag{13}$$

This third approach seems to be the strongest attack, yielding a lower bound of $10^{12}$ MIPS years when $N > 680$. As with the $N$-dimensional lattice, decreasing the norms of $f$ and $g$ does not seem to lower the slope of the line very much, while increasing the norms increases the slope somewhat. A closest vector attack on $(f_1, g_1)$ might decrease this lower bound a bit, but should not alter it substantially.

### 4.5 Forgery via Lattice Reduction

The opponent, Oscar, can try to forge a signature $s$ on a given message $m$ by means of lattice reduction. We show in this section that an ability to accomplish this implies an ability to consistently locate a very short vector in a large class of $(2N+1)$-dimensional lattices.

First consider the case that $D_{\min} = D_{\max} = 0$, so Oscar must find a polynomial $s$ satisfying $s \equiv f_0 * m \pmod{p}$ and such that $t \equiv h * s \pmod{q}$ satisfies $t \equiv g_0 * m \pmod{p}$. Let $m_s$ and $m_t$ be the polynomials with coefficients between $-p/2$ and $p/2$ satisfying $m_s \equiv f_0 * m \bmod p$ and $m_t \equiv g_0 * m \bmod p$, respectively. Consider the $(2N+1)$-dimensional lattice $L_m$ generated by

$$\bigl\{(X^i, X^i * h, 0) : 0 \le i < N\bigr\} \cup \bigl\{(0, qX^i, 0) : 0 \le i < N\bigr\} \cup \bigl\{(m_s, m_t, 1)\bigr\}.$$

Then $L_m$ contains the vector $\tau = (s - m_s, t - m_t, -1)$. The norm of $\tau$ can be estimated by assuming that its coordinates are more-or-less randomly distributed in the interval $[-q/2, q/2]$. This yields $\|\tau\| \approx q\sqrt{N/6}$.

The vector $\tau$ is also contained in the lattice $L_p = (p\mathbb{Z})^{2N} \oplus \mathbb{Z}$. Let $L_{m,p} = L_m \cap L_p$ be the intersection. In other words, letting $I_N$ denote the $N$-by-$N$ identity matrix and $H$ the $N$-by-$N$ circulant matrix formed from the coefficients of the public key $h$, the lattice $L_{m,p}$ is the intersection of the lattices generated by the rows of the following matrices:

$$L_{m,p} = \begin{bmatrix} I_N & H & 0 \\ 0 & qI_N & 0 \\ m_s & m_t & 1 \end{bmatrix} \cap \begin{bmatrix} pI_N & 0 & 0 \\ 0 & pI_N & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Then $L_{m,p}$ has determinant equal to $(\det L)p^{2N}$. Referring to (6) we see that

$$\kappa \approx \sqrt{\pi e q / 6p^2}.$$

For example, $(N, p, q) = (719, 3, 359)$ gives $\kappa \approx 7.5$. This means that the construction of a signed message is equivalent to finding a vector in $L_{m,p}$ that is about 7.5 times longer than the expected shortest vector. It follows that if Oscar is able to forge messages with a reasonable probability, then with reasonable probability he can also find vectors within a factor of 7.5 of the shortest vector. Experiments have indicated that for $N \approx 700$, it requires far in excess of $10^{12}$ MIPS-years to find such a vector in the $(2N+1)$-dimensional lattice $L_{m,p}$. We note also that the probability that such a vector would have all of its coefficients bounded in absolute value by $q/2$ is extremely low.

The case of the optimized parameters of Section 2 is similar. Oscar's best strategy is probably to simply choose $m_s$ at random having the correct properties (i.e., with $\mathrm{Dev}(m_s, f_0 * m)$ in the allowable range) and to choose

$$m_t \equiv g_0 * m \bmod p$$

exactly. The optimized parameters $(N, p, q) = (251, 3, 128)$ lead to a 503-dimensional lattice with $\kappa = 4.5$. Oscar must first try to find a vector no more than 4.5 times longer than the shortest vector. He must then refine his search so that the first $N$ coordinates of his vector have absolute value less than $q/2$ and so that the second $N$ coordinates have at least 55 and no more than 87 coordinates with absolute value greater than $q/2$. The norm condition alone requires about $10^5$ MIPS years for LLL to produce a candidate. Experiments indicate that if the necessary additional constraints are placed on the sup norms of the vectors, then the required time will significantly exceed $10^{12}$ MIPS years.

Another, less efficient, forgery attack requiring a $3N$-dimensional lattice is described in detail in [5].

In conclusion, forgery solutions probably exist in both the general and the optimized versions of NSS, but the time required to find a forgery is sufficiently large so as to preclude a successful attack based on this approach.

### 4.6 Transcript Averaging Attacks

As mentioned previously, examination of a transcript (10) of genuine signatures gives the attacker a sequence of polynomials of the form

$$s \equiv f * w \equiv (f_0 + pf_1)(m + w_1 + pw_2) \pmod{q}$$

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 [5]. 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 [5] for details.

### 4.7 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 4.1 in which one chooses an $m$ and an $s$ with valid $\mathrm{Dev}(s, m)$ and hopes that $t \equiv h*s \pmod q$ has a valid $\mathrm{Dev}(t, g_0*m)$. 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 $m'$.

In the case of zero deviations, if signature encoding is used as suggested in Section 4.9 then it is quite clear that the probability of a successful attack by this method is negligable.

In the case of the optimized parameters the situation is somewhat harder to analyze, but a conservative probabilistic estimate shows that the possibility of a successful forgery is less than $2^{-67}$. For added security, one can 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 less than $2^{-82}$. See [5] for details.

### 4.8 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. For example, recall from Section 4.5 that the existence of a signed message $(m, s)$ implies the existence of a vector in a lattice which is a factor of $\kappa = \sqrt{\pi e q/(6p^2)}$ times larger than the expected smallest vector. We have chosen $p = 3$ for efficiency, but if $p$ is somewhat larger, for fixed $N$, then $\kappa$ will be less than 1. This implies that the existence of such a vector by random chance is extremely unlikely, and that such a vector is probably related to a genuine product $f * w$. If we assume the ability of Oscar to produce such products on demand, given an input $m$, with a somewhat larger $p$ it is not too hard to see that Oscar can probably recover $f_1$.

### 4.9 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 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 $2^{-80}$.

## 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, *Secure user identification based on constrained polynomials*, US Patent 6,076,163, June 13, 2000.
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, J. Pipher, J.H. Silverman, *NSS: A Detailed Analysis of the NTRU Lattice-Based Signature Scheme*, <www.ntru.com>.
6. 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.

7. 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.
8. A.J. Menezes, *Software Implementation of Elliptic Curve Cryptosystems Over Binary Fields*, presentation at CHES 2000, August 17, 2000.
9. A.J. Menezes and P.C. van Oorschot and S.A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1996.
10. I. Mironov, *A note on cryptanalysis of the preliminary version of the NTRU signature scheme*, IACR preprint server, `<http://eprint.iacr.org/2001/005/>`
11. 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.
12. 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.
13. J.H. Silverman. *Estimated Breaking Times for NTRU Lattices*, NTRU Technical Note #012, March 1999, `<www.ntru.com>`.
14. J.H. Silverman. *Almost Inverses and Fast NTRU Key Creation*, NTRU Technical Note #014, March 1999, `<www.ntru.com>`.
15. 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.
16. 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.
17. D. Stinson, *Cryptography: Theory and Practice.* CRC Press, 1997.