Practical lattice-based cryptography: NTRUEncrypt . . .

Chapter · December 2009

DOI: 10.1007/978-3-642-02295-1_11

Nick Howgrave-Graham
University of Bath

Jill Pipher
Brown University

William Whyte
OnBoard Security

Chapter 11
Practical Lattice-Based Cryptography: NTRUEncrypt and NTRUSign

Jeff Hoffstein, Nick Howgrave-Graham, Jill Pipher, and William Whyte

Abstract We provide a brief history and overview of lattice based cryptography and cryptanalysis: shortest vector problems, closest vector problems, subset sum problem and knapsack systems, GGH, Ajtai-Dwork and NTRU. A detailed discussion of the algorithms NTRUEncrypt and NTRUSign follows. These algorithms have attractive operating speed and keysize and are based on hard problems that are seemingly intractable. We discuss the state of current knowledge about the security of both algorithms and identify areas for further research.

Introduction and Overview

In this introduction, we will try to give a brief survey of the uses of lattices in cryptography. Although it is rather a dry way to begin a survey, we should start with some basic definitions related to the subject of lattices. Those with some familiarity with lattices can skip the following section.

Some Lattice Background Material

A lattice L is a discrete additive subgroup of $\mathbb{R}^m$. By discrete, we mean that there exists an $\epsilon > 0$ such that for any $\mathbf{v} \in L$, and all $\mathbf{w} \in \mathbb{R}^m$, if $\|\mathbf{v} - \mathbf{w}\| < \epsilon$, then $\mathbf{w}$ does not belong to the lattice L. This abstract sounding definition transforms into a relatively straightforward reality, and lattices can be described in the following way:

J. Hoffstein (✉)
NTRU Cryptosystems, 35 Nagog Park, Acton, MA 01720, USA,
e-mail: jhoffstein@ntru.com

P.Q. Nguyen and B. Vallée (eds.), *The LLL Algorithm*, Information Security and Cryptography, DOI 10.1007/978-3-642-02295-1_11,

Definition of a lattice

- Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ be a set of vectors in $\mathbb{R}^m$. The set of all linear combinations $a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_k\mathbf{v}_k$, such that each $a_i \in \mathbb{Z}$, is a lattice. We refer to this as the lattice *generated* by $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$.

Bases and the dimension of a lattice

- If $L = \{a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \ldots + a_n\mathbf{v}_n | a_i \in \mathbb{Z}, i = 1, \ldots n\}$ and $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ are n independent vectors, then we say that $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ is a basis for L and that L has dimension n. For any other basis $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k$, we must have $k = n$.

Two different bases for a lattice L are related to each other in almost the same way that two different bases for a vector space V are related to each other. That is, if $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ is a basis for a lattice L then $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n$ is another basis for L if and only if there exist $a_{i,j} \in \mathbb{Z}$ such that

$$\begin{aligned} a_{1,1}\mathbf{v}_1 + a_{1,2}\mathbf{v}_2 + \cdots + \alpha_{1,n}\mathbf{v}_n &= \mathbf{w}_1 \\ a_{2,1}\mathbf{v}_1 + a_{2,2}\mathbf{v}_2 + \cdots + a_{2,n}\mathbf{v}_n &= \mathbf{w}_2 \\ \vdots \qquad & \\ a_{n,1}\mathbf{v}_1 + a_{n,2}\mathbf{v}_2 + \cdots + a_{n,n}\mathbf{v}_n &= \mathbf{w}_n \end{aligned}$$

and the determinant of the matrix

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ & & \vdots & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix}$$

is equal to 1 or -1. The only difference is that the coefficients of the matrix must be integers. The condition that the determinant is nonzero in the vector space case means that the matrix is invertible. This translates in the lattice case to the requirement that the determinant be 1 or -1, the only invertible integers.

A lattice is just like a vector space, except that it is generated by all linear combinations of its basis vectors with integer coefficients, rather than real coefficients. An important object associated to a lattice is the fundamental domain or fundamental parallelepiped. A precise definition is given by:

Let L be a lattice of dimension n with basis $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$. A *fundamental domain* for L corresponding to this basis is

$$\mathcal{F}(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \{t_1\mathbf{v}_1 + t_2\mathbf{v}_2 + \cdots + t_n\mathbf{v}_n \,:\, 0 \le t_i < 1\}.$$

The volume of the fundamental domain is an important invariant associated to a lattice. If L is a lattice of dimension n with basis $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$, the volume of the fundamental domain associated to this basis is called the *determinant* of L and is denoted $\det(L)$.

It is natural to ask if the volume of the fundamental domain for a lattice L depends on the choice of basis. In fact, as was mentioned previously, two different bases for L must be related by an integer matrix W of determinant ± 1. As a result, the integrals measuring the volume of a fundamental domain will be related by a Jacobian of absolute value 1 and will be equal. Thus, the determinant of a lattice is independent of the choice of basis.

Suppose, we are given a lattice L of dimension n. Then, we may formulate the following questions.

1. *Shortest vector problem (SVP)*: Find the shortest non-zero vector in L, i.e., find $0 \neq \mathbf{v} \in L$ such that $\|\mathbf{v}\|$ is minimized.
2. *Closest vector problem (CVP)*: Given a vector $\mathbf{w}$ which is not in L, find the vector $\mathbf{v} \in L$ closest to $\mathbf{w}$, i.e., find $\mathbf{v} \in L$ such that $\|\mathbf{v} - \mathbf{w}\|$ is minimized.

Both of these problems appear to be profound and very difficult as the dimension n becomes large. Solutions, or even partial solutions to these problems also turn out to have surprisingly many applications in a number of different fields. In full generality, the CVP is known to be NP-hard and SVP is NP-hard under a certain "randomized reduction" hypothesis.[1] Also, SVP is NP-hard when the norm or distance used is the l^∞ norm. In practice, a CVP can often be reduced to a SVP and is thought of as being "a little bit harder" than SVP. Reduction of CVP to SVP is used by in [2] to prove that SVP is hard in Ajtai's probabilistic sense. The interested reader can consult Micciancio's book [3] for a more compete treatment of the complexity of lattice problems. In practice it is very hard to achieve "full generality." In a real world scenario, a cryptosystem based on an NP-hard or NP-complete problem may use a particular subclass of that problem to achieve efficiency. It is then possible that this subclass of problems could be easier to solve than the general problem.

Secondary problems, that are also very important, arise from SVP and CVP. For example, one could look for a basis $\mathbf{v}_1, \ldots, \mathbf{v}_n$ of L consisting of all "short" vectors (e.g., minimize $\max \|\mathbf{v}_i\|$). This is known as the Short Basis Problem or SBP. Alternatively, one might search for a nonzero vector $\mathbf{v} \in L$ satisfying

$$\|\mathbf{v}\| \leq \psi(n) \|\mathbf{v}_{\text{shortest}}\|,$$

where ψ is some slowly growing function of n, the dimension of L. For example, for a fixed constant κ, one could try to find $\mathbf{v} \in L$ satisfying

$$\|\mathbf{v}\| \leq \kappa \sqrt{n} \|\mathbf{v}_{\text{shortest}}\|,$$

and similarly for CVP. These generalizations are known as approximate shortest and closest vector problems, or ASVP, ACVP.

[1] Under this hypothesis, the class of polynomial time algorithms is enlarged to include those that are not deterministic but will with high probability terminate in polynomial time. See Ajtai [1]

BookID 183387_ChapID 011_Proof# 1 - 02/10/09

How big, in fact, is the shortest vector in terms of the determinant and the dimension of L? A theorem of Hermite from the nineteenth century says that for a fixed dimension n there exists a constant γ_n so that in every lattice L of dimension n, the shortest vector satisfies

$$\|\mathbf{v}_{\text{shortest}}\|^2 \leq \gamma_n \det(L)^{2/n}.$$

Hermite showed that $\gamma_n \leq (4/3)^{(n-1)/2}$. The smallest possible value one can take for γ_n is called *Hermite's constant*. Its exact value is known only for $1 \leq n \leq 8$ and for $n = 24$ [4]. For example, $\gamma_2 = \sqrt{4/3}$. We now explain why, for large n, Hermite's constant should be no larger than $\mathcal{O}(n)$.

Although exact bounds for the size of the shortest vector of a lattice are unknown for large n, one can make probabilistic arguments using the Gaussian heuristic. One variant of the Gaussian heuristic states that for a fixed lattice L and a sphere of radius r centered at 0, as r tends to infinity, the ratio of the volume of the sphere divided by $\det L$ will approach the number of points of L inside the sphere. In two dimensions, if L is simply $\mathbb{Z}^2$, the question of how precisely the area of a circle approximates the number of integer points inside the circle is a classical problem in number theory. In higher dimensions, the problem becomes far more difficult. This is because as n increases the error created by lattice points near the surface of the sphere can be quite large. This becomes particularly problematic for small values of r. Still, one can ask the question: For what value of r does the ratio

$$\frac{\mathrm{Vol}(S)}{\det L}$$

approach 1. This gives us in some sense an expected value for r, the smallest radius at which the expected number of points of L with length less than r equals 1. Performing this computation and using Stirling's formula to approximate factorials, we find that for large n this value is approximately

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

For this reason, we make the following definition:

If L is a lattice of dimension n, we define the Gaussian expected shortest length to be

$$\sigma(L) = \sqrt{\frac{n}{2\pi e}}\,(\det(L))^{1/n}.$$

We will find this value $\sigma(L)$ to be useful in quantifying the difficulty of locating short vectors in lattices. It can be thought of as the probable length of the shortest vector of a "random" lattice of given determinant and dimension. It seems to be the case that if the actual shortest vector of a lattice L is significantly shorter than $\sigma(L)$, then LLL and related algorithms have an easier time locating the shortest vector.

A heuristic argument identical to the above can be used to analyze the CVP. Given a vector $\mathbf{w}$ which is not in L, we again expect a sphere of radius r centered about $\mathbf{w}$ to contain one point of L after the radius is such that the volume of the sphere equals $\det(L)$. In this case also, the CVP becomes easier to solve as the ratio of actual distance to the closest vector of L over "expected distance" decreases.

Knapsacks

The problems of factoring integers and finding discrete logarithms are believed to be difficult since no one has yet found a polynomial time algorithm for producing a solution. One can formulate the decision form of the factoring problem as follows: does there exist a factor of N less than p? This problem belongs to NP and another complexity class, co-NP. Because it is widely believed that NP is not the same as co-NP, it is also believed that factoring is not an NP-complete problem. Naturally, a cryptosystem whose underlying problem is known to be NP-hard would inspire greater confidence in its security. Therefore, there has been a great deal of interest in building efficient public key cryptosystems based on such problems. Of course, the fact that a certain problem is NP-hard does not mean that every instance of it is NP-hard, and this is one source of difficulty in carrying out such a program.

The first such attempt was made by Merkle and Hellman in the late 70s [5], using a particular NP-complete problem called the subset sum problem. This is stated as follows:

> **The subset sum problem**
>
> Suppose one is given a list of positive integers $\{M_1, M_2, \ldots, M_n\}$. An unknown subset of the list is selected and summed to give an integer S. Given S, recover the subset that summed to S, or find another subset with the same property.

Here, there is another way of describing this problem. A list of positive integers $\mathbf{M} = \{M_1, M_2, \ldots, M_n\}$ is public knowledge. Choose a secret binary vector $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, where each x_i can take on the value 1 or 0. If

$$S = \sum_{i=1}^{n} x_i M_i$$

then how can one recover the original vector $\mathbf{x}$ in an efficient way? (Of course, there might also be another vector $\mathbf{x}'$ which also gives S when dotted with $\mathbf{M}$.)

The difficulty in translating the subset sum problem into a cryptosystem is that of building in a trapdoor. Merkle and Hellman's system took advantage of the fact that there are certain subset sum problems that are extremely easy to solve. Suppose that one takes a sequence of positive integers $\mathbf{r} = \{r_1, r_2, \ldots, r_n\}$ with the property that $r_{i+1} \geq 2r_i$ for each $1 \leq i \leq n$. Such a sequence is called *super increasing*. Given an integer S, with $S = \mathbf{x} \cdot \mathbf{r}$ for a binary vector $\mathbf{x}$, it is easy to recover $\mathbf{x}$ from S.

The basic idea that Merkle and Hellman proposed was this: begin with a secret super increasing sequence $\mathbf{r}$ and choose two large secret integers A, B, with $B > 2r_n$ and $(A, B) = 1$. Here, r_n is the last and largest element of $\mathbf{r}$, and the lower bound condition ensures that B must be larger than any possible sum of a subset of the r_i. Multiply the entries of $\mathbf{r}$ by A and reduce modulo B to obtain a new sequence $\mathbf{M}$, with each $M_i \equiv Ar_i \pmod{B}$. This new sequence $\mathbf{M}$ is the public key. Encryption then works as follows. The message is a secret binary vector $\mathbf{x}$ which is encrypted to $S = \mathbf{x} \cdot \mathbf{M}$. To decrypt S, multiply by $A^{-1} \pmod{B}$ to obtain $S' \equiv \mathbf{x} \cdot \mathbf{r} \pmod{B}$. If S' is chosen in the range $0 \le S' \le B - 1$, one obtains an exact inequality $S' = \mathbf{x} \cdot \mathbf{r}$, as any subset of the integers r_i must sum to an integer smaller than B. The sequence $\mathbf{r}$ is super increasing and $\mathbf{x}$ may be recovered.

A cryptosystem of this type is known as a *knapsack system*. The general idea is to start with a secret super increasing sequence, disguise it by some collection of modular linear operations, then reveal the transformed sequence as the public key. The original Merkle and Hellman system suggested applying a secret permutation to the entries of $A\mathbf{r} \pmod{B}$ as an additional layer of security. Later versions were proposed by a number of people, involving multiple multiplications and reductions with respect to various moduli. For an excellent survey, see the article by Odlyzko [6].

The first question one must ask about a knapsack system is that what minimal properties must $\mathbf{r}$, A, and B have to obtain a given level of security? Some very easy attacks are possible if r_1 is too small, so one generally takes $2^n < r_1$. But, what is the minimal value of n that we require? Because of the super increasing nature of the sequence, one has

$$r_n = \mathcal{O}(S) = \mathcal{O}(2^{2n}).$$

The space of all binary vectors $\mathbf{x}$ of dimension n has size 2^n , and thus an exhaustive search for a solution would require effort on the order of 2^n. In fact, a meet in the middle attack is possible, thus the security of a knapsack system with a list of length n is $O(2^{n/2})$.

While the message consists of n bits of information, the public key is a list of n integers, each approximately $2n$ bits long and there requires about $2n^2$ bits. Therefore, taking $n = 160$ leads to a public key size of about 51200 bits. Compare this to RSA or Diffie-Hellman, where, for security on the order of 2^{80}, the public key size is about 1000 bits.

The temptation to use a knapsack system rather than RSA or Diffie-Hellman was very great. There was a mild disadvantage in the size of the public key, but decryption required only one (or several) modular multiplications and none were required to encrypt. This was far more efficient than the modular exponentiations in RSA and Diffie-Hellman.

Unfortunately, although a meet in the middle attack is still the best known attack on the general subset sum problem, there proved to be other, far more effective, attacks on knapsacks with trapdoors. At first, some very specific attacks were announced by Shamir, Odlyzko, Lagarias, and others. Eventually, however, after the publication of the famous LLL paper [7] in 1985, it became clear that a secure knapsack-based system would require the use of an n that was too large to be practical.

A public knapsack can be associated to a certain lattice L as follows. Given a public list $\mathbf{M}$ and encrypted message S, one constructs the matrix

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & m_1 \\ 0 & 1 & 0 & \cdots & 0 & m_2 \\ 0 & 0 & 1 & \cdots & 0 & m_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & m_n \\ 0 & 0 & 0 & \cdots & 0 & S \end{pmatrix}$$

with row vectors $\mathbf{v}_1 = (1, 0, 0, \ldots, 0, m_1), \mathbf{v}_2 = (0, 1, 0, \ldots, 0, m_2), \ldots, \mathbf{v}_n = (0, 0, 0, \ldots, 1, m_n)$ and $\mathbf{v}_{n+1} = (0, 0, 0, \ldots, 0, S)$. The collection of all linear combinations of the $\mathbf{v}_i$ with integer coefficients is the relevant lattice L. The determinant of L equals S. The statement that the sum of some subset of the m_i equals S translates into the statement that there exists a vector $\mathbf{t} \in L$,

$$\mathbf{t} = \sum_{i=1}^{n} x_i \mathbf{v}_i - \mathbf{v}_{n+1} = (x_1, x_2, \ldots, x_n, 0),$$

where each x_i is chosen from the set $\{0, 1\}$. Note that the last entry in $\mathbf{t}$ is 0 because the subset sum problem is solved and the sum of a subset of the m_i is canceled by the S.

The crux of the matter

As the x_i are binary, $\|\mathbf{t}\| \leq \sqrt{n}$. In fact, as roughly half of the x_i will be equal to 0, it is very likely that $\|\mathbf{t}\| \approx \sqrt{n/2}$. On the other hand, the size of each $\|\mathbf{v}_i\|$ varies between roughly 2^n and 2^{2n}. The key observation is that it seems rather improbable that a linear combination of vectors that are so large should have a norm that is so small.

The larger the weights m_i were, the harder the subset sum problem was to solve by combinatorial means. Such a knapsack was referred to as a *low density* knapsack. However, for low density knapsacks, S was larger and thus the ratio of the actual smallest vector to the expected smallest vector was smaller. Because of this, the LLL lattice reduction method was more more effective on a low density knapsack than on a generic subset sum problem.

It developed that, using LLL, if n is less than around 300, a secret message $\mathbf{x}$ can be recovered from an encrypted message S in a fairly short time. This meant that in order to have even a hope of being secure, a knapsack would need to have $n > 300$, and a corresponding public key length that was greater than 180000 bits. This was sufficiently impractical that knapsacks were abandoned for some years.

Expanding the Use of LLL in Cryptanalysis

Attacks on the discrete logarithm problem and factorization were carefully analyzed and optimized by many researchers, and their effectiveness was quantified. Curiously, this did not happen with LLL, and improvements in lattice reduction methods such as BKZ that followed it. Although quite a bit of work was done on improving lattice reduction techniques, the precise effectiveness of these techniques on lattices of various characteristics remained obscure. Of particular interest was the question of how the running times of LLL and BKZ required to solve SVP or CVP varied with the dimension of the lattice, the determinant, and the ratio of the actual shortest vector's length to the expected shortest length.

In 1996–1997, several cryptosystems were introduced whose underlying hard problem was SVP or CVP in a lattice L of dimension n. These were, in alphabetical order:

- Ajtai-Dwork, ECCC report 1997 [8]
- GGH, presented at Crypto '97 [9]
- NTRU, presented at the rump session of Crypto '96 [10]

The public key sizes associated to these cryptosystems were $\mathcal{O}(n^4)$ for Ajtai-Dwork, $\mathcal{O}(n^2)$ for GGH, and $\mathcal{O}(n \log n)$ for NTRU.

The system proposed by Ajtai and Dwork was particularly interesting in that they showed that it was provably secure unless a worst case lattice problem could be solved in polynomial time. Offsetting this, however, was the large key size. Subsequently, Nguyen and Stern showed, in fact, that any efficient implementation of the Ajtai-Dwork system was insecure [11].

The GGH system can be explained very simply. The owner of the private key has the knowledge of a special small, reduced basis R for L. A person wishing to encrypt a message has access to the public key B, which is a generic basis for L. The basis B is obtained by multiplying R by several random unimodular matrices, or by putting R into Hermite normal form, as suggested by Micciancio.

We associate to B and R, corresponding matrices whose rows are the n vectors in the respective basis. A plaintext is a row vector of n integers, $\mathbf{x}$, and the encryption of $\mathbf{x}$ is obtained by computing $\mathbf{e} = \mathbf{x}B + \mathbf{r}$, where $\mathbf{r}$ is a random perturbation vector consisting of small integers. Thus, $\mathbf{x}B$ is contained in the lattice L while $\mathbf{e}$ is not. Nevertheless, if $\mathbf{r}$ is short enough, then with high probability, $\mathbf{x}B$ is the unique point in L which is closest to $\mathbf{e}$.

A person with knowledge of the private basis R can compute $\mathbf{x}B$ using Babai's technique [12], from which $\mathbf{x}$ is then obtained. More precisely, using the matrix R, one can compute $\mathbf{e}R^{-1}$ and then round each coefficient of the result to the nearest integer. If $\mathbf{r}$ is sufficiently small, and R is sufficiently short and close to being orthogonal, then the result of this rounding process will most likely recover the point $\mathbf{x}B$.

Without the knowledge of any reduced basis for L, it would appear that breaking GGH was equivalent to solving a general CVP. Goldreich, Goldwasser, and Halevi conjectured that for $n > 300$ this general CVP would be intractable. However, the effectiveness of LLL (and later variants of LLL) on lattices of high dimension had not been closely studied. In [13], Nguyen showed that some information leakage in GGH encryption allowed a reduction to an easier CVP problem, namely one where the ratio of actual distance to the closest vector to expected length of the shortest vector of L was smaller. Thus, he was able to solve GGH challenge problems in dimensions 200, 250, 300, and 350. He did not solve their final problem in dimension 400, but at that point the key size began to be too large for this system to be practical. It also was not clear at this point how to quantify the security of the $n = 400$ case.

The NTRU system was described at the rump session of Crypto '96 as a ring based public key system that could be translated into an SVP problem in a special class of lattices.[2] Specifically, the NTRU lattice L consists of all integer row vectors of the form $(\mathbf{x}, \mathbf{y})$ such that

$$\mathbf{y} \equiv \mathbf{x}H \pmod{q}.$$

Here, q is a public positive integer, on the order of 8 to 16 bits, and H is a public circulant matrix. Congruence of vectors modulo q is interpreted component-wise. Because of its circulant nature, H can be described by a single vector, explaining the shorter public keys.

An NTRU private key is a single short vector $(\mathbf{f}, \mathbf{g})$ in L. This vector is used, rather than Babai's technique, to solve a CVP for decryption. Together with its rotations, $(\mathbf{f}, \mathbf{g})$ yields half of a reduced basis. The vector $(\mathbf{f}, \mathbf{g})$ is likely to be the shortest vector in the public lattice, and thus NTRU is vulnerable to efficient lattice reduction techniques.

At Eurocrypt '97, Coppersmith and Shamir pointed out that any sufficiently short vector in L, not necessarily $(\mathbf{f}, \mathbf{g})$ or one of its rotations, could be used as a decryption key. However, they remarked that this really did not matter as:

"We believe that for recommended parameters of the NTRU cryptosystem, the LLL algorithm will be able to find the original secret key $\mathbf{f}$..."

However, no evidence to support this belief was provided, and the very interesting question of quantifying the effectiveness of LLL and its variants against lattices of NTRU type remained.

At the rump session of Crypto '97, Lieman presented a report on some preliminary work by himself and the developers of NTRU on this question. This report, and many other experiments supported the assertion that the time required for LLL-BKZ to find the smallest vector in a lattice of dimension n was at least exponential in n. See [14] for a summary of part of this investigation.

The original algorithm of LLL corresponds to block size 2 of BKZ and provably returns a reasonably short vector of the lattice L. The curious thing is that in low dimensions this vector tends to be the actual shortest vector of L. Experiments have led us to the belief that the BKZ block size required to find the actual shortest vector

[2] NTRU was published in ANTS '98. Its appearance in print was delayed by its rejection by the Crypto '97 program committee.

in a lattice is linear in the dimension of the lattice, with an implied constant depending upon the ratio of the actual shortest vector length over the Gaussian expected shortest length. This constant is sufficiently small that in low dimensions the relevant block size is 2. It seems possible that it is the smallness of this constant that accounts for the early successes of LLL against knapsacks. The exponential nature of the problem overcomes the constant as n passes 300.

Digital Signatures Based on Lattice Problems

In general, it is very straight forward to associate a digital signature process to a lattice where the signer possess a secret highly reduced basis and the verifier has only a public basis for the same lattice. A message to be signed is sent by some public hashing process to a random point $\mathbf{m}$ in $\mathbb{Z}^n$. The signer, using the method of Babai and the private basis, solves the CVP and finds a lattice point $\mathbf{s}$ which is reasonably close to $\mathbf{m}$. This is the signature on the message $\mathbf{m}$. Anyone can verify, using the public basis, that $\mathbf{s} \in L$ and $\mathbf{s}$ is close to $\mathbf{m}$. However, presumably someone without the knowledge of the reduced basis would have a hard time finding a lattice point $\mathbf{s}'$ sufficiently close to $\mathbf{m}$ to count as a valid signature.

However, any such scheme has a fundamental problem to overcome: every valid signature corresponds to a vector difference $\mathbf{s} - \mathbf{m}$. A transcript of many such $\mathbf{s} - \mathbf{m}$ will be randomly and uniformly distributed inside a fundamental parallelepiped of the lattice. This counts as a leakage of information and as Nguyen and Regev recently showed, this vulnerability makes any such scheme subject to effective attacks based on independent component analysis [15].

In GGH, the private key is a full reduced basis for the lattice, and such a digital signature scheme is straightforward to both set up and attack. In NTRU, the private key only reveals half of a reduced basis, making the process of setting up an associated digital signature scheme considerably less straightforward.

The first attempt to base a digital signature scheme upon the same principles as "NTRU encryption" was NSS [16]. Its main advantage, (and also disadvantage) was that it relied *only* on the information immediately available from the private key, namely half of a reduced basis. The incomplete linkage of the NSS signing process to the CVP problem in a full lattice required a variety of ad hoc methods to bind signatures and messages, which were subsequently exploited to break the scheme. An account of the discovery of the fatal weaknesses in NSS can be found in Sect. 7 of the extended version of [17], available at [18].

This paper contains the second attempt to base a signature scheme on the NTRU lattice (NTRUSign) and also addresses two issues. First, it provides an algorithm for generating the full short basis of an NTRU lattice from the knowledge of the private key (half the basis) and the public key (the large basis). Second, it described a method of perturbing messages before signing to reduce the efficiency of transcript leakage (see Section "NTRUSign Signature Schemes: Perturbations"). The learning theory approach of Nguyen and Regev in [15] shows that about 90,000 signatures compromises the security of basic NTRUSign without perturbations. W. Whyte pointed out at the rump session of Crypto '06 that by applying rotations to effectively increase the number of signatures, the number of signatures required to theoretically determine a private key was only about 1000. Nguyen added this approach to his and Regev's technique and was able to, in fact, recover the private key with roughly this number of signatures.

The NTRUEncrypt and NTRUSign Algorithms

The rest of this article is devoted to a description of the NTRUEncrypt and NTRUSign algorithms, which at present seem to be the most efficient embodiments of public key algorithms whose security rests on lattice reduction.

NTRUEncrypt

NTRUEncrypt is typically described as a polynomial based cryptosystem involving convolution products. It can naturally be viewed as a lattice cryptosystem too, for a certain restricted class of lattices.

The cryptosystem has several natural parameters and, as with all practical cryptosystems, the hope is to optimize these parameters for efficiency while at the same time avoiding all known cryptanalytic attacks.

One of the more interesting cryptanalytic techniques to date concerning NTRUEncrypt exploits the property that, under certain parameter choices, the cryptosystem can fail to properly decrypt valid ciphertexts. The *functionality* of the cryptosystem is not adversely affected when these, so-called, "decryption failures" occur with only a very small probability on random messages, but an attacker can choose messages to induce failure, and assuming he knows when messages have failed to decrypt (which is a typical security model in cryptography) there are efficient ways to extract the private key from knowledge of the failed ciphertexts (i.e., the decryption failures are highly key-dependent). This was first noticed in [19, 20] and is an important consideration in choosing parameters for NTRUEncrypt.

Other security considerations for NTRUEncrypt parameters involve assessing the security of the cryptosystem against lattice reduction, meet-in-the-middle attacks based on the structure of the NTRU private key, and hybrid attacks that combine both of these techniques.

NTRUSign

The search for a "zero-knowledge" lattice-based signature scheme is a fascinating open problem in cryptography. It is worth commenting that most cryptographers would assume that anything purporting to be a signature scheme would automatically have the property of "zero-knowledge," i.e., the definition of a signature scheme implies the problems of determining the private key or creating forgeries should become not easier after having seen a polynomial number of valid signatures. However, in the theory of lattices, signature schemes with reduction arguments are just emerging and their computational effectiveness is currently being examined. For most lattice-based signature schemes, there are explicit attacks known which use the knowledge gained from a transcript of signatures.

When considering *practical signature schemes*, the "zero-knowledge" property is not essential for the scheme to be useful. For example, smart cards typically burn out before signing a million times, so if the private key in infeasible to obtain (and a forgery is impossible to create) with a transcript of less than a million signatures, then the signature scheme would be sufficient in this environment. It, therefore, seems that there is value in developing efficient, non-zero-knowledge, lattice-based signature schemes.

The early attempts [16, 21] at creating such practical signature schemes from NTRU-based concepts succumbed to attacks which required transcripts of far too small a size [22, 23]. However, the known attacks on NTRUSign, the currently recommended, signature scheme, require transcript lengths of impractical length, i.e., the signatures scheme does appear to be of practical significance at present.

NTRUSign was invented between 2001 and 2003 by the inventors of NTRUEncrypt together with N. Howgrave-Graham and W. Whyte [17]. Like NTRUEncrypt it is highly parametrizable and, in particular, has a parameter involving the number of perturbations. The most interesting cryptanalytic progress on NTRUSign has been showing that it *must* be used with at least one perturbation, i.e., there is an efficient and elegant attack [15, 24] requiring a small transcript of signatures in the case of zero perturbations.

Contents and Motivation

This paper presents an overview of operations, performance, and security considerations for NTRUEncrypt and NTRUSign. The most up-to-date descriptions of NTRUEncrypt and NTRUSign are included in [25] and [26], respectively. This paper summarizes, and draws heavily on, the material presented in those papers.

This paper is structured as follows. First, we introduce and describe the algorithms NTRUEncrypt and NTRUSign. We then survey known results about the security of these algorithms, and then present performance characteristics of the algorithms.

As mentioned above, the motivation for this work is to produce viable cryptographic primitives based on the theory of lattices. The benefits of this are twofold: the new schemes may have operating characteristics that fit certain environments particularly well. Also, the new schemes are based on different hard problems from the current mainstream choices of RSA and ECC.

The second point is particularly relevant in a post-quantum world. Lattice reduction is a reasonably well-studied hard problem that is currently not known to be solved by any polynomial time, or even subexponential time, quantum algorithms [27, 28]. While the algorithms are definitely of interest even in the classical computing world, they are clearly prime candidates for widespread adoption should quantum computers ever be invented.

NTRUEncrypt: Overview

Parameters and Definitions

An implementation of the NTRUEncrypt encryption primitive is specified by the following parameters:

- N — *Degree Parameter*. A positive integer. The associated NTRU lattice has dimension $2N$.
- q — *Large Modulus*. A positive integer. The associated NTRU lattice is a convolution modular lattice of modulus q.
- p — *Small Modulus*. An integer or a polynomial.
- $\mathcal{D}_f, \mathcal{D}_g$ — *Private Key Spaces*. Sets of small polynomials from which the private keys are selected.
- $\mathcal{D}_m$ — *Plaintext Space*. Set of polynomials that represent encryptable messages. It is the responsibility of the encryption scheme to provide a method for encoding the message that one wishes to encrypt into a polynomial in this space.
- $\mathcal{D}_r$ — *Blinding Value Space*. Set of polynomials from which the temporary blinding value used during encryption is selected.
- `center` — *Centering Method*. A means of performing mod q reduction on decryption.

Definition 1. The *Ring of Convolution Polynomials* is

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

Multiplication of polynomials in this ring corresponds to the convolution product of their associated vectors, defined by

$$(\mathsf{f} * \mathsf{g})(X) = \sum_{k=0}^{N-1} \Bigg(\sum_{i+j \equiv k \pmod N} \mathsf{f}_i \cdot \mathsf{g}_j \Bigg) X^k \ .$$

We also use the notation $\mathcal{R}_q = \frac{(\mathbb{Z}/q\mathbb{Z})[X]}{(X^N-1)}$. Convolution operations in the ring $\mathcal{R}_q$ are referred to as *modular convolutions*.

Definition 2. A polynomial $a(X) = a_0 + a_1 X + \cdots + a_{N-1} X^{N-1}$ is identified with its vector of coefficients $\mathsf{a} = [a_0, a_1, \ldots, a_{N-1}]$. The mean $\bar{\mathsf{a}}$ of a polynomial a is defined by $\bar{a} = \frac{1}{N} \sum_{i=0}^{N-1} \mathsf{a}_i$. The *centered norm* $\|\mathsf{a}\|$ of a is defined by

$$\|\mathsf{a}\|^2 = \sum_{i=0}^{N-1} a_i^2 - \frac{1}{N} \left(\sum_{i=0}^{N-1} a_i \right)^2 . \tag{11.1}$$

Definition 3. The *width* $\mathsf{Width}(\mathsf{a})$ of a polynomial or vector is defined by

$$\mathsf{Width}(\mathsf{a}) = \mathsf{Max}(a_0, \ldots, a_{N-1}) - \mathsf{Min}(a_0, \ldots, a_{N-1}) .$$

Definition 4. A *binary polynomial* is one whose coefficients are all in the set $\{0, 1\}$. A *trinary polynomial* is one whose coefficients are all in the set $\{0, \pm 1\}$. If one of the inputs to a convolution is a binary polynomial, the operation is referred to as a *binary convolution*. If one of the inputs to a convolution is a trinary polynomial, the operation is referred to as a *trinary convolution*.

Definition 5. Define the polynomial spaces $\mathcal{B}_N(d), \mathcal{T}_N(d), \mathcal{T}_N(d_1, d_2)$ as follows. Polynomials in $\mathcal{B}_N(d)$ have d coefficients equal to 1, and the other coefficients are 0. Polynomials in $\mathcal{T}_N(d)$ have $d+1$ coefficients equal to 1, have d coefficients equal to -1, and the other coefficients are 0. Polynomials in $\mathcal{T}_N(d_1, d_2)$ have d_1 coefficients equal to 1, have d_2 coefficients equal to -1, and the other coefficients are 0.

"Raw" NTRUEncrypt

Key Generation

NTRUEncrypt *key generation* consists of the following operations:

1. Randomly generate polynomials f and g in $\mathcal{D}_f$, $\mathcal{D}_g$, respectively.
2. Invert f in $\mathcal{R}_q$ to obtain f_q, invert f in $\mathcal{R}_p$ to obtain f_p, and check that g is invertible in $\mathcal{R}_q$ [29].
3. The public key $\mathsf{h} = p * \mathsf{g} * \mathsf{f}_q \pmod q$. The private key is the pair $(\mathsf{f}, \mathsf{f}_p)$.

Encryption

NTRUEncrypt *encryption* consists of the following operations:

1. Randomly select a "small"polynomial $\mathsf{r} \in \mathcal{D}_r$.
2. Calculate the ciphertext e as $\mathsf{e} \equiv \mathsf{r} * \mathsf{h} + \mathsf{m} \pmod q$.

Decryption

NTRUEncrypt *decryption* consists of the following operations:

1. Calculate $\mathsf{a} \equiv \texttt{center}(\mathsf{f} * e)$, where the centering operation `center` reduces its input into the interval $[A, A+q-1]$.
2. Recover m by calculating $\mathsf{m} \equiv \mathsf{f}_p * \mathsf{a} \pmod{p}$.

To see why decryption works, use $\mathsf{h} \equiv p * \mathsf{g} * \mathsf{f}_q$ and $\mathsf{e} \equiv \mathsf{r} * \mathsf{h} + \mathsf{m}$ to obtain

$$\mathsf{a} \equiv p * \mathsf{r} * \mathsf{g} + \mathsf{f} * \mathsf{m} \pmod{q}\ . \tag{11.2}$$

For appropriate choices of parameters and `center`, this is an equality over $\mathbb{Z}$, rather than just over $\mathbb{Z}_q$. Therefore, step 2 recovers m: the $p * \mathsf{r} * \mathsf{g}$ term vanishes, and $\mathsf{f}_p * \mathsf{f} * \mathsf{m} = \mathsf{m} \pmod{p}$.

Encryption Schemes: NAEP

To protect against adaptive chosen ciphertext attacks, we must use an appropriately defined *encryption scheme*. The scheme described in [30] gives provable security in the random oracle model [31, 32] with a tight (i.e., linear) reduction. We briefly outline it here.

NAEP uses two hash functions:

$$G : \{0,1\}^{N-l} \times \{0,1\}^l \to \mathcal{D}_r \qquad H : \{0,1\}^N \to \{0,1\}^N$$

To encrypt a message $M \in \{0,1\}^{N-l}$ using NAEP one uses the functions

$$\texttt{compress}(x) = (x \pmod{q}) \pmod{2},$$
$$\texttt{B2P} : \{0,1\}^N \to \mathcal{D}_m \cup \text{“error”}, \quad \texttt{P2B} : \mathcal{D}_m \to \{0,1\}^N$$

The function `compress` puts the coefficients of the modular quantity $x \pmod{q}$ in to the interval $[0, q)$, and then this quantity is reduced modulo 2. The role of `compress` is simply to reduce the size of the input to the hash function H for gains in practical efficiency.The function `B2P` converts a bit string into a binary polynomial, or returns “error” if the bit string does not fulfil the appropriate criteria – for example, if it does not have the appropriate level of combinatorial security. The function `P2B` converts a binary polynomial to a bit string.

The encryption algorithm is then specified by:

1. Pick $b \xleftarrow{R} \{0,1\}^l$.
2. Let $\mathsf{r} = G(M, b)$, $\mathsf{m} = \texttt{B2P}(\ (M||b) \oplus H(\texttt{compress}(\mathsf{r} * \mathsf{h}))\ )$.
3. If `B2P` returns “error”, go to step 1.
4. Let $\mathsf{e} = \mathsf{r} * \mathsf{h} + \mathsf{m} \in \mathcal{R}_q$.

Step 3 ensures that only messages of the appropriate form will be encrypted.

To decrypt a message $\mathsf{e} \in \mathcal{R}_q$, one does the following:

1. Let $\mathsf{a} = \mathtt{center}(\mathsf{f} * \mathsf{e} \pmod{q})$.
2. Let $\mathsf{m} = \mathsf{f}_p^{-1} * \mathsf{a} \pmod{p}$.
3. Let $\mathsf{s} = \mathsf{e} - \mathsf{m}$.
4. Let $M||b = \mathtt{P2B}(\mathsf{m}) \oplus H(\mathtt{compress}(\mathtt{P2B}(\mathsf{s})))$.
5. Let $\mathsf{r} = G(M, b)$.
6. If $\mathsf{r} * \mathsf{h} = \mathsf{s} \pmod{q}$, and $\mathsf{m} \in \mathcal{D}_m$, then return the message M, else return the string "invalid ciphertext."

The use of the scheme NAEP introduces a single additional parameter:

l *Random Padding Length*. The length of the random padding b concatenated with M in step 1.

Instantiating NAEP*:* SVES-3

The EESS#1 v2 standard [21] specifies an instantiation of NAEP known as SVES-3. In SVES-3, the following specific design choices are made:

- To allow variable-length messages, a one-byte encoding of the message length in bytes is prepended to the message. The message is padded with zeroes to fill out the message block.
- The hash function G which is used to produce r takes as input M; b; an OID identifying the encryption scheme and parameter set; and a string h_{trunc} derived by truncating the public key to length l_h bits.

SVES-3 includes h_{trunc} in G so that r depends on the specific public key. Even if an attacker was to find an (M, b) that gave an r with an increased chance of a decryption failure, that (M, b) would apply only to a single public key and could not be used to attack other public keys. However, the current recommended parameter sets do not have decryption failures and so there is no need to input h_{trunc} to G. We will therefore use SVES-3but set $l_h = 0$.

NTRUEncrypt *Coins!*

It is both amusing and informative to view the NTRUEncrypt operations as working with "coins." By coins, we really mean N-sided coins, like the British 50 pence piece.

An element of $\mathcal{R}$ maps naturally to an N-sided coin: one simply write the integer entries of $a \in \mathcal{R}$ on the side-faces of the coin (with "heads" facing up, say). Multiplication by X in $\mathcal{R}$ is analagous to simply rotating the coin, and addition of two elements in $\mathcal{R}$ is analagous to placing the coins on top of each other and summing the faces. A generic multiplication by an element in $\mathcal{R}$ is thus analagous to multiple copies of the same coin being rotated by different amonuts, placed on top of each other, and summed.

The NTRUEncrypt key recovery problem is a binary multiplication problem, i.e., given d_f copies of the h-coin the problem is to pile them on top of eachother (with distinct rotations) so that the faces sum to zero or one modulo q.

The raw NTRUEncrypt encryption function has a similar coin analogy: one piles d_r copies of the h-coin on top of one another with random (but distinct) rotations, then one sums the faces modulo q, and adds a small $\{0, 1\}$ perturbation to faces modulo q (corresponding to the message). The resulting coin, c, is a valid NTRUEncrypt ciphertext.

The NTRUEncrypt decryption function also has a similar coin analogy: one piles d_f copies of a c-coin (corresponding to the ciphertext) on top of each other with rotations corresponding to f. After summing the faces modulo q, centering, and then a reduction modulo p, one should recover the original message m.

These NTRUEncrypt operations are so easy, it seems strong encryption could have been used centuries ago, had public-key encryption been known about. From a number theoretic point of view, the only nontrivial operation is the creation of the h coin (which involves Euclid's algorithm over polynomials).

NTRUSign: Overview

Parameters

An implementation of the NTRUSign primitive uses the following parameters:

N	Polynomials have degree $< N$
q	Coefficients of polynomials are reduced modulo q
$\mathcal{D}_f, \mathcal{D}_g$	Polynomials in $\mathcal{T}(d)$ have $d+1$ coefficients equal to 1, have d coefficients equal to -1, and the other coefficients are 0.
$\mathcal{N}$	The norm bound used to verify a signature.
β	The balancing factor for the norm $\|\cdot\|_\beta$. Has the property $0 < \beta \le 1$.

"Raw" NTRUSign

Key Generation

NTRUSign *key generation* consists of the following operations:

1. Randomly generate "small" polynomials f and g in $\mathcal{D}_f$, $\mathcal{D}_g$, respectively, such that f and g are invertible modulo q.

2. Find polynomials F and G such that

$$\mathsf{f} * \mathsf{G} - \mathsf{g} * \mathsf{F} = q\ , \tag{11.3}$$

   and F and G have size

$$\|\mathsf{F}\| \approx \|\mathsf{G}\| \approx \|\mathsf{f}\|\sqrt{N/12}\ . \tag{11.4}$$

   This can be done using the methods of [17]
3. Denote the inverse of f in $\mathcal{R}_q$ by f_q, and the inverse of g in $\mathcal{R}_q$ by g_q. The public key $\mathsf{h} = \mathsf{F} * \mathsf{f}_q \pmod q = \mathsf{G} * \mathsf{g}_q \pmod q$. The private key is the pair (f, g).

Signing

The signing operation involves *rounding* polynomials. For any $a \in \mathbb{Q}$, let $\lfloor a \rceil$ denote the integer closest to a, and define $\{a\} = a - \lfloor a \rceil$. (For numbers a that are midway between two integers, we specify that $\{a\} = +\frac{1}{2}$, rather than $-\frac{1}{2}$.) If A is a polynomial with rational (or real) coefficients, let $\lfloor A \rceil$ and $\{A\}$ be A with the indicated operation applied to each coefficient.

"Raw" NTRUSign *signing* consists of the following operations:

1. Map the digital document D to be signed to a vector $\mathsf{m} \in [0, q)^N$ using an agreed hash function.
2. Set

$$(\mathsf{x}, \mathsf{y}) = (0, \mathsf{m}) \begin{pmatrix} \mathsf{G} & -\mathsf{F} \\ -\mathsf{g} & \mathsf{f} \end{pmatrix} / q = \left(\frac{-\mathsf{m} * \mathsf{g}}{q}, \frac{\mathsf{m} * \mathsf{f}}{q} \right).$$

3. Set

$$\epsilon = -\{x\} \qquad \text{and} \qquad \epsilon' = -\{y\}\ . \tag{11.5}$$

4. Calculate s, the signature, as

$$\mathsf{s} = \epsilon \mathsf{f} + \epsilon' \mathsf{g}\ . \tag{11.6}$$

Verification

Verification involves the use of a *balancing factor* β and a *norm bound* $\mathcal{N}$. To verify, the recipient does the following:

1. Map the digital document D to be verified to a vector $\mathsf{m} \in [0, q)^N$ using the agreed hash function.
2. Calculate $\mathsf{t} = \mathsf{s} * \mathsf{h} \bmod q$, where s is the signature, and h is the signer's public key.

3. Calculate the norm

$$\nu = \min_{k_1,k_2 \in R} \left(\|\mathsf{s} + k_1 q\|^2 + \beta^2 \|(\mathsf{t} - \mathsf{m}) + k_2 q\|^2 \right)^{1/2}. \tag{11.7}$$

4. If $\nu \leq \mathcal{N}$, the verification succeeds, otherwise, it fails.

Why NTRUSign *Works*

Given any positive integers N and q and any polynomial $h \in R$, we can construct a lattice L_h contained in $R^2 \cong \mathbb{Z}^{2N}$ as follows:

$$L_h = L_h(N,q) = \left\{ (r, r') \in R \times R \;\middle|\; r' \equiv r * h \pmod{q} \right\}.$$

This sublattice of $\mathbb{Z}^{2N}$ is called a *convolution modular lattice*. It has dimension equal to $2N$ and determinant equal to q^N.

Since

$$\det \begin{pmatrix} \mathsf{f} & \mathsf{F} \\ \mathsf{g} & \mathsf{G} \end{pmatrix} = q$$

and we have defined $\mathsf{h} = \mathsf{F}/\mathsf{f} = \mathsf{G}/\mathsf{g} \bmod q$, we know that

$$\begin{pmatrix} \mathsf{f} & \mathsf{F} \\ \mathsf{g} & \mathsf{G} \end{pmatrix} \text{and} \begin{pmatrix} 1 & \mathsf{h} \\ 0 & q \end{pmatrix}$$

are bases for the same lattice. Here, as in [17], a 2-by-2 matrix of polynomials is converted to a $2N$-by-$2N$ integer matrix matrix by converting each polynomial in the polynomial matrix to its representation as an N-by-N circulant matrix, and the two representations are regarded as equivalent.

Signing consists of finding a close lattice point to the message point $(0, \mathsf{m})$ using Babai's method: express the target point as a real-valued combination of the basis vectors, and find a close lattice point by rounding off the fractional parts of the real coefficients to obtain integer combinations of the basis vectors. The error introduced by this process will be the sum of the rounding errors on each of the basis vectors, and the rounding error by definition will be between $-\frac{1}{2}$ and $\frac{1}{2}$. In NTRUSign, the basis vectors are all of the same length, so the expected error introduced by $2N$ roundings of this type will be $\sqrt{N/6}$ times this length.

In NTRUSign, the private basis is chosen such that $\|\mathsf{f}\| = \|\mathsf{g}\|$ and $\|\mathsf{F}\| \sim \|\mathsf{G}\| \sim \sqrt{N/12}\|\mathsf{f}\|$. The expected error in signing will therefore be

$$\sqrt{N/6}\|\mathsf{f}\| + \beta(N/6\sqrt{2})\|\mathsf{f}\|. \tag{11.8}$$

In contrast, an attacker who uses only the public key will likely produce a signature with N incorrect coefficients, and those coefficients will be distributed randomly mod q. The expected error in generating a signature with a public key is therefore

$$\beta\sqrt{N/12}q\ . \tag{11.9}$$

(We discuss security considerations in more detail in Section "NTRUSign Security Considerations" and onwards; the purpose of this section is to argue that it is plausible that the private key allows the production of smaller signatures than the public key).

It is therefore clear that it is possible to choose $\|\mathsf{f}\|$ and q such that the knowledge of the private basis allows the creation of smaller signing errors than knowledge of the public basis alone. Therefore, by ensuring that the signing error is less than that could be expected to be produced by the public basis, a recipient can verify that the signature was produced by the owner of the private basis and is therefore valid.

NTRUSign *Signature Schemes: Chosen Message Attacks, Hashing, and Message Preprocessing*

To prevent chosen message attacks, the message representative m must be generated in some pseudo-random fashion from the input document D. The currently recommended hash function for NTRUSign is a simple Full Domain Hash. First the message is hashed to a "seed" hash value H_m. H_m is then hashed in counter mode to produce the appropriate number of bits of random output, which are treated as N numbers mod q. Since q is a power of 2, there are no concerns with bias.

The above mechanism is deterministic. If parameter sets were chosen that gave a significant chance of signature failure, the mechanism can be randomized as follows. The additional input to the process is r_{len}, the length of the randomizer in bits.

On signing:

1. Hash the message as before to generate H_m.
2. Select a randomizer r consisting of r_{len} random bits.
3. Hash $H_m\|r$ in counter mode to obtain enough output for the message representative m.
4. On signing, check that the signature will verify correctly.

   a. If the signature does not verify, repeat the process with a different r.
   b. If the signature verifies, send the tuple (r, s) as the signature.

On verification, the verifier uses the received r and the calculated H_m as input to the hash in counter mode to generate the same message representative as the signer used.

The size of r should be related to the probability of signature failure. An attacker who is able to determine through timing information that a given H_m required multiple rs knows that at least one of those rs resulted in a signature that was too big, but does not know which message it was or what the resulting signature was. It is an open research question to quantify the appropriate size of r for a given signature failure probability, but in most cases, $r_{\text{len}} = 8$ or 32 should be sufficient.

NTRUSign *Signature Schemes: Perturbations*

To protect against transcript attacks, the raw NTRUSign signing algorithm defined above is modified as follows.

On key generation, the signer generates a secret *perturbation distribution function*.

On signing, the signer uses the agreed hash function to map the document D to the message representative m. However, before using his or her private key, he or she chooses an error vector e drawn from the perturbation distribution function that was defined as part of key generation. He or she then signs $\mathsf{m} + \mathsf{e}$, rather than m alone.

The verifier calculates m, t, and the norms of s and $\mathsf{t}-\mathsf{m}$ and compares the norms to a specified bound $\mathcal{N}$ as before. Since signatures with perturbations will be larger than unperturbed signatures, $\mathcal{N}$ and, in fact, all of the parameters will in general be different for the perturbed and unpertubed cases.

NTRU currently recommends the following mechanism for generating perturbations.

Key Generation

At key generation time, the signer generates B lattices $L_1 \ldots L_B$. These lattices are generated with the same parameters as the private and public key lattice, L_0, but are otherwise independent of L_0 and of each other. For each L_i, the signer stores f_i, g_i, h_i.

Signing

When signing m, for each L_i starting with L_B, the signer does the following:

1. Set $(\mathsf{x}, \mathsf{y}) = \left(\frac{-\mathsf{m}*\mathsf{g}_i}{q}, \frac{\mathsf{m}*\mathsf{f}_i}{q}\right)$.
2. Set $\epsilon = -\{x\}$ and $\epsilon' = -\{y\}$.
3. Set $\mathsf{s}_i = \epsilon \mathsf{f}_i + \epsilon' \mathsf{g}_i$.
4. Set $\mathsf{s} = \mathsf{s} + \mathsf{s}_i$.
5. If $i = 0$ stop and output s; otherwise, continute
6. Set $\mathsf{t}_i = \mathsf{s}_i * \mathsf{h}_i \mod q$
7. Set $\mathsf{m} = \mathsf{t}_i - (\mathsf{s}_i * \mathsf{h}_{i-1}) \mod q$.

The final step translates back to a point of the form $(0, \mathsf{m})$ so that all the signing operations can use only the f and g components, allowing for greater efficiency. Note that steps 6 and 7 can be combined into the single step of setting $\mathsf{m} = \mathsf{s}_i * (\mathsf{h}_i - \mathsf{h}_{i-1})$ to improve performance.

The parameter sets defined in [26] take $B = 1$.

NTRUEncrypt Performance

NTRUEncrypt *Parameter Sets*

There are many different ways of choosing "small" polynomials. This section reviews NTRU's current recommendations for choosing the form of these polynomials for the best efficiency. We focus here on choices that improve efficiency; security considerations are looked at in Section "NTRUEncrypt Security Considerations".

Form of f

Published NTRUEncrypt parameter sets [25] take f to be of the form $\mathsf{f} = 1 + p\mathsf{F}$. This guarantees that $\mathsf{f}_p = 1$, eliminating one convolution on decryption.

Form of F, g, r

NTRU currently recommends several different forms for F and r. If F and r take *binary* and *trinary* form, respectively, they are drawn from $\mathcal{B}_N(d)$, the set of binary polynomials with d 1s and $N - d$ 0s or $\mathcal{T}_N(d)$, the set of trinary polynomials with $d+1$ 1s, d -1s and $N-2d-1$ 0s. If F and r take *product* form, then $\mathsf{F} = \mathsf{f}_1 * \mathsf{f}_2 + \mathsf{f}_3$, with $\mathsf{f}_1, \mathsf{f}_2, \mathsf{f}_3 \stackrel{R}{\leftarrow} \mathcal{B}_N(d), \mathcal{T}_N(d)$, and similarly for r. (The value d is considerably lower in the product-form case than in the binary or trinary case).

A binary or trinary convolution requires on the order of dN adds mod q. The best efficiency is therefore obtained when d is as low as possible consistent with the security requirements.

Plaintext Size

For k-bit security, we want to transport $2k$ bits of message and we require $l \geq k$, l the random padding length. SVES-3 uses 8 bits to encode the length of the transported message. N must therefore be at least $3k + 8$. Smaller N will in general lead to lower bandwidth and faster operations.

Form of p, q

The parameters p and q must be relatively prime. This admits of various combinations, such as $(p = 2, q = \text{prime})$, $(p = 3, q = 2^m)$, and $(p = 2 + X, q = 2^m)$.

The `B2P` Function

The polynomial m produced by the `B2P` function will be a random trinary polynomial. As the number of 1s, (in the binary case), or 1s and -1s (in the trinary case), decreases, the strength of the ciphertext against both lattice and combinatorial attacks will decrease. The `B2P` function therefore contains a check that the number of 1s in m is not less than a value d_{m_0}. This value is chosen to be equal to df. If, during encryption, the encrypter generates m that does not satisfy this criterion, they must generate a different value of b and re-encrypt.

NTRUEncrypt *Performance*

Table 11.1 and Table 11.2 give parameter sets and running times (in terms of operations per second) for size optimized and speed optimized performance, respectively, at different security levels corresponding to k bits of security. "Size" is the size of the public key in bits. In the case of NTRUEncrypt and RSA, this is also the size of the ciphertext; in the case of some ECC encryption schemes, such as ECIES, the ciphertext may be a multiple of this size. Times given are for unoptimized C implementations on a 1.7 GHz Pentium and include time for all encryption scheme operations, including hashing, random number generation, as well as the primitive operation. d_{m_0} is the same in both the binary and product-form case and is omitted from the product-form table.

For comparison, we provide the times given in [33] for raw elliptic curve point multiplication (not including hashing or random number generation times) over the

Table 11.1 Size-optimized NTRUEncrypt parameter sets with trinary polynomials

k	N	d	d_{m_0}	q	size	RSA size	ECC size	enc/s	dec/s	ECC mult/s	Enc ECC ratio	Dec ECC ratio
112	401	113	113	2,048	4,411	2,048	224	2,640	1,466	1,075	4.91	1.36
128	449	134	134	2,048	4,939	3,072	256	2,001	1,154	661	6.05	1.75
160	547	175	175	2,048	6,017	4,096	320	1,268	718	n/a	n/a	n/a
192	677	157	157	2,048	7,447	7,680	384	1,188	674	196	12.12	3.44
256	1,087	120	120	2,048	11,957	15, 360	512	1,087	598	115	18.9	5.2

Table 11.2 Speed-optimized NTRUEncrypt parameter sets with trinary polynomials

k	N	d	d_{m_0}	q	$size$	RSA size	ECC size	enc/s	dec/s	ECC mult/s	Enc ECC ratio	Dec ECC ratio
112	659	38	38	2,048	7, 249	2,048	224	4,778	2,654	1,075	8.89	2.47
128	761	42	42	2,048	8, 371	3,072	256	3,767	2,173	661	11.4	3.29
160	991	49	49	2048	10, 901	4,096	320	2,501	1,416	n/a	n/a	n/a
192	1, 087	63	63	2,048	11, 957	7,680	384	1,844	1,047	196	18.82	5.34
256	1, 499	79	79	2,048	16, 489	15,360	512	1,197	658	115	20.82	5.72

NIST prime curves. These times were obtained on a 400 MHz SPARC and have been converted to operations per second by simply scaling by $400/1700$. Times given are for point multiplication without precomputation, as this corresponds to common usage in encryption and decryption. Precomputation improves the point multiplication times by a factor of 3.5–4. We also give the speedup for NTRUEncrypt decryption vs. a single ECC point multiplication.

NTRUSign Performance

NTRUSign *Parameter Sets*

Form of f, g

The current recommended parameter sets take f and g to be trinary, i.e., drawn from $\mathcal{T}_N(d)$. Trinary polynomials allow for higher combinatorial security than binary polynomials at a given value of N and admit efficient implementations. A trinary convolution requires $(2d + 1)N$ adds and one subtract mod q. The best efficiency is therefore obtained when d is as low as possible consistent with the security requirements.

Form of p, q

The parameters q and N must be relatively prime. For efficiency, we take q to be a power of 2.

Signing Failures

A low value of $\mathcal{N}$, the norm bound, gives the possibility that a validly generated signature will fail. This affects efficiency, as if the chance of failure is non-negligible, the signer must randomize the message before signing and check for failure on signature generation. For efficiency, we want to set $\mathcal{N}$ sufficiently high to make the chance of failure negligible. To do this, we denote the expected size of a signature by $\mathcal{E}$ and define the *signing tolerance* ρ by the formula

$$\mathcal{N} = \rho\mathcal{E} .$$

As ρ increases beyond 1, the chance of a signing failure appears to drop off exponentially. In particular, experimental evidence indicates that the probability that a validly generated signature will fail the normbound test with parameter ρ is smaller than $e^{-C(N)(\rho-1)}$, where $C(N) > 0$ increases with N. In fact, under the assumption that each coefficient of a signature can be treated as a sum of independent identically distributed random variables, a theoretical analysis indicates that $C(N)$ grows quadratically in N. The parameter sets below were generated with $\rho = 1.1$, which appears to give a vanishingly small probability of valid signature failure for N in the ranges that we consider. It is an open research question to determine precise signature failure probabilities for specific parameter sets, i.e., to determine the constants in $C(N)$.

NTRUSign *Performance*

With one perturbation, signing takes time equivalent to two "raw" signing operations (as defined in Section "Signing") and one verification. Research is ongoing into alternative forms for the perturbations that could reduce this time.

Table 11.3 gives the parameter sets for a range of security levels, corresponding to k-bit security, and the performance (in terms of signatures and verifications per second) for each of the recommended parameter sets. We compare signature times to a single ECC point multiplication with precomputation from [33]; without precomputation, the number of ECC signatures/second goes down by a factor of 3.5–4. We compare verification times to ECDSA verification times without memory constraints from [33]. As in Tables 11.1 and 11.2, NTRUSign times given are for the entire scheme (including hashing, etc.), not just the primitive operation, while ECDSA times are for the primitive operation alone.

Above the 80-bit security level, NTRUSign signatures are smaller than the corresponding RSA signatures. They are larger than the corresponding ECDSA signatures by a factor of about 4. An NTRUSign private key consists of sufficient space to store f and g for the private key, plus sufficient space to store f_i, g_i, and h_i for each of the B perturbation bases. Each f and g can be stored in $2N$ bits, and each h can be stored in $N \log_2(q)$ bits, so the total storage required for the one-perturbation

Table 11.3 Performance measures for different NTRUSign parameter sets. (Note: parameter sets have not been assessed against the hybrid attack of Section "The Hybrid Attack" and may give less than k bits of security)

Parameters				Public key and				sign/s			vfy/s		
k	N	d	q	NTRU	ECDSA key	ECDSA sig	RSA	NTRU	ECDSA	Ratio	NTRU	ECDSA	Ratio
80	157	29	256	1,256	192	384	1,024	4,560	5,140	0.89	15,955	1,349	11.83
112	197	28	256	1,576	224	448	~2,048	3,466	3,327	1.04	10,133	883	11.48
128	223	32	256	1,784	256	512	3,072	2,691	2,093	1.28	7,908	547	14.46
160	263	45	512	2,367	320	640	4,096	1,722	–	–	5,686	–	–
192	313	50	512	2,817	384	768	7,680	1,276	752	1.69	4,014	170	23.61
256	349	75	512	3,141	512	1024	15,360	833	436	1.91	3,229	100	32.29

case is $16N$ bits for the 80- to 128-bit parameter sets below and $17N$ bits for the 160- to 256-bit parameter sets, or approximately twice the size of the public key.

Security: Overview

We quantify security in terms of bit strength k, evaluating how much effort an attacker has to put in to break a scheme. All the attacks we consider here have variable running times, so we describe the strength of a parameter set using the notion of *cost*. For an algorithm $\mathcal{A}$ with running time t and probability of success ε, the cost is defined as

$$C_{\mathcal{A}} = t/\varepsilon \ .$$

This definition of cost is not the only one that could be used. For example, in the case of indistinguishability against adaptive chosen-ciphertext attack, the attacker outputs a single bit $i \in \{0, 1\}$, and obviously has a chance of success of at least $\frac{1}{2}$. Here, the probability of success is less important than the attacker's *advantage*, defined as

$$\mathrm{adv}(\mathcal{A}(\mathsf{ind})) = 2.(\mathbb{P}[\mathsf{Succ}[\mathcal{A}]] - 1/2) \ .$$

However, in this paper, the cost-based measure of security is appropriate.

Our notion of cost is derived from [34] and related work. An alternate notion of cost, which is the definition above multiplied by the amount of memory used, is proposed in [35]. The use of this measure would allow significantly more efficient parameter sets, as the meet-in-the-middle attack described in Section "Combinatorial Security" is essentially a time-memory tradeoff that keeps the product of time and memory constant. However, current practice is to use the measure of cost above.

We also acknowledge that the notion of comparing public-key security levels with symmetric security levels, or of reducing security to a single headline measure, is inherently problematic – see an attempt to do so in [36], and useful comments on this in [37]. In particular, extrapolation of breaking times is an inexact science, the behavior of breaking algorithms at high security levels is by definition untested, and one can never disprove the existence of an algorithm that attacks NTRUEncrypt (or any other system) more efficiently than the best currently known method.

Common Security Considerations

This section deals with security considerations that are common to NTRUEncrypt and NTRUSign.

Most public key cryptosystems, such as RSA [38] or ECC [39, 40], are based on a one-way function for which there is one best-known method of attack: factoring in the case of RSA, Pollard-rho in the case of ECC. In the case of NTRU, there are *two* primary methods of approaching the one-way function, both of which must be considered when selecting a parameter set.

Combinatorial Security

Polynomials are drawn from a known space $\mathcal{S}$. This space can best be searched by using a combinatorial technique originally due to Odlyzko [41], which can be used to recover f or g from h or r and m from e. We denote the combinatorial security of polynomials drawn from $\mathcal{S}$ by Comb[$\mathcal{S}$]

$$\mathrm{Comb}[\mathcal{B}_N(d)] \geq \frac{\binom{N/2}{d/2}}{\sqrt{N}}\,. \tag{11.10}$$

For trinary polynomials in $\mathcal{T}_N(d)$, we find

$$\mathrm{Comb}[\mathcal{T}(d)] > \binom{N}{d+1}/\sqrt{N}. \tag{11.11}$$

For product-form polynomials in $\mathcal{P}_N(d)$, defined as polynomials of the form $\mathsf{a} = \mathsf{a}_1 * \mathsf{a}_2 + \mathsf{a}_3$, where $\mathsf{a}_1, \mathsf{a}_2, \mathsf{a}_3$ are all binary with $d_{a_1}, d_{a_2}, d_{a_3}$ 1s respectively, $d_{a1} = d_{a2} = d_{a3} = d_a$, and there are no further constraints on a, we find [25]:

$$\begin{aligned}\mathrm{Comb}[\mathcal{P}_N(d)] \geq \min\Bigg(&\binom{N - \lceil N/d \rceil}{d-1}^2, \\ &\max\left(\binom{N - \lceil \frac{N}{d} \rceil}{d-1}\binom{N - \lceil \frac{N}{d-)} \rceil}{d-2}, \binom{N}{2d}\right), \\ &\max\left(\binom{N}{d}\binom{N}{d-1}, \binom{N - \lceil \frac{N}{2d} \rceil}{2d-1}\right)\Bigg)\end{aligned}$$

Lattice Security

An NTRU public key h describes a $2N$-dimensional NTRU lattice containing the private key (f, g) or (f, F). When f is of the form $\mathsf{f} = 1 + p\mathsf{F}$, the best lattice attack on the private key involves solving a Close Vector Problem (CVP).[3] When f is not of the

[3] Coppersmith and Shamir [42] propose related approaches which turn out not to materially affect security.

form $\mathsf{f} = 1 + p\mathsf{F}$, the best lattice attack involves solving an Approximate Shortest Vector Problem (apprSVP). Experimentally, it has been found that an NTRU lattice of this form can usefully be characterized by two quantities

$$
\begin{aligned}
a &= N/q, \\
c &= \sqrt{4\pi e \|\mathsf{F}\|\|\mathsf{g}\|/q} \quad \text{(NTRUEncrypt)}, \\
&= \sqrt{4\pi e \|\mathsf{f}\|\|\mathsf{F}\|/q} \quad \text{(NTRUSign)}.
\end{aligned}
$$

(For product-form keys the norm $\|\mathsf{F}\|$ is variable but always obeys $|\mathsf{F}| \geq \sqrt{D(N-D)/N}$, $D = d^2 + d$. We use this value in calculating the lattice security of product-form keys, knowing that in practice the value of c will typically be higher.)

This is to say that for constant (a, c), the experimentally observed running times for lattice reduction behave roughly as

$$\log(T) = AN + B\ ,$$

for some experimentally-determined constants A and B.

Table 11.4 summarizes experimental results for breaking times for NTRU lattices with different (a, c) values. We represent the security by the constants A and B. The breaking time in terms of bit security is $AN + B$. It may be converted to time in MIPS-years using the equality 80 bits$\sim 10^{12}$ MIPS-years.

For constant (a, c), increasing N increases the breaking time exponentially. For constant (a, N), increasing c increases the breaking time. For constant (c, N), increasing a decreases the breaking time, although the effect is slight. More details on this table are given in [14].

Note that the effect of moving from the "standard" NTRUEncrypt lattice to the "transpose" NTRUSign lattice is to increase c by a factor of $(N/12)^{1/4}$. This allows for a given level of lattice security at lower dimensions for the transpose lattice than for the standard lattice. Since NTRUEncrypt uses the standard lattice, NTRUEncrypt key sizes given in [25] are greater than the equivalent NTRUSign key sizes at the same level of security.

The technique known as *zero-forcing* [14,43] can be used to reduce the dimension of an NTRU lattice problem. The precise amount of the expected performance gain is heavily dependent on the details of the parameter set; we refer the reader to [14, 43] for more details. In practice, this reduces security by about 6–10 bits.

Table 11.4 Extrapolated bit security constants depending on (c, a)

c	a	A	B
1.73	0.53	0.3563	−2.263
2.6	0.8	0.4245	−3.440
3.7	2.7	0.4512	+0.218
5.3	1.4	0.6492	−5.436

The Hybrid Attack

In this section, we will review the method of [44]. The structure of the argument is simpler for the less efficient version of NTRU where the public key has the form $h \equiv f^{-1} * g \pmod q$. The rough idea is as follows. Suppose one is given N, q, d, e, h and hence implicitly an NTRUEncrypt public lattice L of dimension $2N$. The problem is to locate the short vector corresponding to the secret key (f, g). One first chooses $N_1 < N$ and removes a $2N_1$ by $2N_1$ lattice L_1 from the center of L. Thus, the original matrix corresponding to L has the form

$$\left(\begin{array}{c|c} qI_N & 0 \\ \hline H & I_N \end{array}\right) = \left(\begin{array}{c|c|c} qI_{N-N_1} & 0 & 0 \\ \hline * & L_1 & 0 \\ \hline * & * & I_{N-N_1} \end{array}\right) \tag{11.12}$$

and L_1 has the form

$$\left(\begin{array}{c|c} qI_{N_1} & 0 \\ \hline H_1 & I_{N_1} \end{array}\right). \tag{11.13}$$

Here, H_1 is a truncated piece of the circulant matrix H corresponding to h appearing in (11.12). For increased flexibility, the upper left and lower right blocks of L_1 can be of different sizes, but for ease of exposition, we will consider only the case where they are equal.

Let us suppose that an attacker must use a minimum of k_1 bits of effort to reduce L_1 until all N_1 of the q-vectors are removed. When this is done and L_1 is put in lower triangular form, the entries on the diagonal will have values $\{q^{\alpha_1}, q^{\alpha_2}, \ldots, q^{\alpha_{2N_1}}\}$, where $\alpha_1 + \cdots + \alpha_{2N_1} = N_1$, and the α_i will come very close to decreasing linearly, with

$$1 \approx \alpha_1 > \cdots > \alpha_{2N_1} \approx 0.$$

That is to say, L_1 will roughly obey the geometric series assumption or GSA. This reduction will translate back to a corresponding reduction of L, which when reduced to lower triangular form will have a diagonal of the form

$$\{q, q, \ldots, q, q^{\alpha_1}, q^{\alpha_2}, \ldots, q^{\alpha_{2N_1}}, 1, 1, \ldots, 1\}.$$

The key point here is that it requires k_1 bits of effort to achieve this reduction, with $\alpha_{2N_1} \approx 0$. If $k_2 > k_1$ bits are used, then the situation can be improved to achieve $\alpha_{2N_1} = \alpha > 0$. As k_2 increases the value of α is increased.

In the previous work, the following method was used to launch the meet in the middle attack. It was assumed that the coefficients of f are partitioned into two blocks. These are of size N_1 and $K = N - N_1$. The attacker guesses the coefficients of f that fall into the K block and then uses the reduced basis for L to check if his or her guess is correct. The main observation of [44] is that a list of guesses can be made about half the coefficients in the K block and can be compared to a list of guesses about the other half of the coefficients in the K block. With a probability $p_s(\alpha)$, a correct matching of two half guesses can be confirmed, where $p_s(0) = 0$ and $p_s(\alpha)$ increases monotonically with α. In [44], a value of $\alpha = 0.182$ was used with a corresponding probability $p_s(0.182) = 2^{-13}$. The probability $p_s(0.182)$ was computed by sampling and the bit requirement, k_2 was less than 60.3. In general, if one used k_2 bits of lattice reduction work to obtain a given $p_s(\alpha)$ (as large as possible), then the number of bits required for a meet in the middle search through the K block decreases as K decreases and as $p_s(\alpha)$ increases.

A very subtle point in [44] was the question of how to optimally choose N_1 and k_2. The objective of an attacker was to choose these parameters so that k_2 equalled the bit strength of a meet in the middle attack on K, given the $p_s(\alpha)$ corresponding to N_1. It is quite hard to make an optimal choice, and for details we refer the reader to [44] and [45].

One Further Remark

For both NTRUEncrypt and NTRUSign the degree parameter N must be prime. This is because, as Gentry observed in [46], if N is the composite, the related lattice problem can be reduced to a similar problem in a far smaller dimension. This reduced problem is then comparatively easy to solve.

NTRUEncrypt **Security Considerations**

Parameter sets for NTRUEncrypt at a k-bit security level are selected subject to the following constraints:

- The work to recover the private key or the message through lattice reduction must be at least k bits, where bits are converted to MIPS-years using the equality 80 bits $\sim 10^{12}$ MIPS-years.
- The work to recover the private key or the message through combinatorial search must be at least 2^k binary convolutions.
- The chance of a decryption failure must be less than 2^{-k}.

Decryption Failure Security

NTRU decryption can fail on validly encrypted messages if the `center` method returns the wrong value of A, or if the coefficients of $\mathsf{prg} + \mathsf{fm}$ do not lie in an interval of width q. Decryption failures leak information about the decrypter's private key [19, 20]. The recommended parameter sets ensure that decryption failures will not happen by setting q to be greater than the maximum possible width of $\mathsf{prg} + \mathsf{m} + \mathsf{pFm}$. q should be as small as possible while respecting this bound, as lowering q increases the lattice constant c and hence the lattice security. Centering then becomes simply a matter of reducing into the interval $[0, q-1]$.

It would be possible to improve performance by relaxing the final condition to require only that the probability of a decryption failure was less than 2^{-K}. However, this would require improved techniques for estimating decryption failure probabilities.

N, q, and p

The small and large moduli p and q must be relatively prime in the ring $\mathcal{R}$. Equivalently, the three quantities

$$p, \quad q, \quad X^N - 1$$

must generate the unit ideal in the ring $\mathbb{Z}[X]$. (As an example of why this is necessary, in the extreme case that p divides q, the plaintext is equal to the ciphertext reduced modulo p.)

Factorization of $X^N - 1$ (mod q)

If $\mathsf{F}(X)$ is a factor of $X^N - 1 \pmod{q}$, and if $\mathsf{h}(X)$ is a multiple of $\mathsf{F}(X)$, i.e., if $\mathsf{h}(X)$ is zero in the field $K = (\mathbb{Z}/q\mathbb{Z})[X]/\mathsf{F}(X)$, then an attacker can recover the value of $\mathsf{m}(X)$ in the field K.

If q is prime and has order $t \pmod{N}$, then

$$X^N - 1 \equiv (X-1)\mathsf{F}_1(X)\mathsf{F}_2(X)\cdots\mathsf{F}_{(N-1)/t}(X) \quad \text{in } (\mathbb{Z}/q\mathbb{Z})[X],$$

where each $\mathsf{F}_i(X)$ has degree t and is irreducible mod q. (If q is the composite, there are corresponding factorizations.) If $\mathsf{F}_i(X)$ has degree t, the probability that $\mathsf{h}(X)$ or $\mathsf{r}(X)$ is divisible by $\mathsf{F}_i(X)$ is presumably $1/q^t$. To avoid attacks based on the factorization of h or r, we will require that for each prime divisor P of q, the order of $P \pmod{N}$ must be $N-1$ or $(N-1)/2$. This requirement has the useful side-effect of increasing the probability that randomly chosen f will be invertible in $\mathcal{R}_q$ [47].

Information Leakage from Encrypted Messages

The transformation $\mathsf{a} \to \mathsf{a}(1)$ is a ring homomorphism, and so the ciphertext e has the property that

$$\mathsf{e}(1) = \mathsf{r}(1)\mathsf{h}(1) + \mathsf{m}(1)\,.$$

An attacker will know $\mathsf{h}(1)$, and for many choices of parameter set $\mathsf{r}(1)$ will also be known. Therefore, the attacker can calculate $\mathsf{m}(1)$. The larger $|\mathsf{m}(1) - N/2|$ is, the easier it is to mount a combinatorial or lattice attack to recover the msssage, so the sender should always ensure that $\|\mathsf{m}\|$ is sufficiently large. In these parameter sets, we set a value d_{m_0} such that there is a probability of less than 2^{-40} that the number of 1s or 0s in a randomly generated m is less than d_{m_0}. We then calculate the security of the ciphertext against lattice and combinatorial attacks in the case where m has exactly this many 1s and require this to be greater than 2^k for k bits of security.

NTRUEncrypt *Security: Summary*

In this section, we present a summary of the security measures for the parameter sets under consideration. Table 11.5 gives security measures optimized for size. Table 11.6 gives security measures optimized for speed. The parameter sets for NTRUEncrypt have been calculated based on particular conservative assumptions about the effectiveness of certain attacks. In particular, these assumptions assume the attacks will be improved in certain ways over the current best known attacks, although we do not know yet exactly how these improvements will be implemented. The tables below show the strength of the current recommended parameter sets against the best attacks that are currently known. As attacks improve, it will be instructive to watch the "known hybrid strength" reduce to the recommended security level. The "basic lattice strength" column measures the strength against a pure lattice-based (nonhybrid) attack.

NTRUSign **Security Considerations**

This section considers security considerations that are specific to NTRUSign.

Table 11.5 NTRUEncrypt security measures for size-optimized parameters using trinary polynomials

Recommended security level	N	q	d_f	Known hybrid strength	c	Basic lattice strength
112	401	2,048	113	154.88	2.02	139.5
128	449	2,048	134	179.899	2.17	156.6
160	547	2,048	175	222.41	2.44	192.6
192	677	2,048	157	269.93	2.5	239
256	1,087	2,048	120	334.85	2.64	459.2

Table 11.6 NTRUEncrypt security measures for speed-optimized parameters using trinary polynomials

Recommended security level	N	q	d_f	Known hybrid strength	c	Basic lattice strength
112	659	2,048	38	137.861	1.74	231.5
128	761	2,048	42	157.191	1.85	267.8
160	991	2,048	49	167.31	2.06	350.8
192	1,087	2,048	63	236.586	2.24	384
256	1,499	2,048	79	312.949	2.57	530.8

Security Against Forgery

We quantify the probability that an adversary, without the knowledge of f, g, can compute a signature s on a given document D. The constants $N, q, \delta, \beta, \mathcal{N}$ must be chosen to ensure that this probability is less than 2^{-k}, where k is the desired bit level of security. To investigate this, some additional notation will be useful:

1. Expected length of s: $\mathcal{E}_s$
2. Expected length of $t - m$: $\mathcal{E}_t$

By $\mathcal{E}_s$, $\mathcal{E}_t$, we mean, respectively, the expected values of $\|s\|$ and $\|t - m\|$ (appropriately reduced mod$q$) when generated by the signing procedure described in Section "Signing". These will be independent of m but dependent on N, q, δ. A genuine signature will then have expected length

$$\mathcal{E} = \sqrt{\mathcal{E}_s^2 + \beta^2 \mathcal{E}_t^2}$$

and we will set

$$\mathcal{N} = \rho \sqrt{\mathcal{E}_s^2 + \beta^2 \mathcal{E}_t^2}. \tag{11.14}$$

As in the case of recovering the private key, an attack can be made by combinatorial means, by lattice reduction methods or by some mixing of the two. By balancing these approaches, we will determine the optimal choice of β, the public scaling factor for the second coordinate.

Combinatorial Forgery

Let us suppose that $N, q, \delta, \beta, \mathcal{N}, h$ are fixed. An adversary is given m, the image of a digital document D under the hash function H. His or her problem is to locate an s such that

$$\|(s \bmod q, \beta(h * s - m) \bmod q)\| < \mathcal{N}.$$

In particular, this means that for an appropriate choice of $k_1, k_2 \in R$

$$(\|(s + k_1 q\|^2 + \beta^2 \|h * s - m + k_2 q)\|^2)^{1/2} < \mathcal{N}.$$

A purely combinatorial attack that the adversary can take is to choose s at random to be quite small, and then to hope that the point $h * s - m$ lies inside of a sphere of radius $\mathcal{N}/\beta$ about the origin after its coordinates are reduced mod$q$. The attacker can also attempt to combine guesses. Here, the attacker would calculate a series of random s_i and the corresponding t_i and $t_i - m$, and file the t_i and the $t_i - m$ for future reference. If a future s_j produces a t_j that is sufficiently close to $t_i - m$, then $(s_i + s_j)$ will be a valid signature on m. As with the previous meet-in-the-middle attack, the core insight is that filing the t_i and looking for collisions allow us to check l^2 t-values while generating only l s-values.

An important element in the running time of attacks of this type is the time that it takes to file a t value. We are interested not in exact collisions, but in two t_i that lie close enough to allow forgery. In a sense, we are looking for a way to file the t_i in a spherical box, rather than in a cube as is the case for the similar attacks on private keys. It is not clear that this can be done efficiently. However, for safety, we will assume that the process of filing and looking up can be done in constant time, and that the running time of the algorithm is dominated by the process of searching the s-space. Under this assumption, the attacker's expected work before being able to forge a signature is:

$$p(N, q, \beta, \mathcal{N}) < \sqrt{\frac{\pi^{N/2}}{\Gamma(1 + N/2)} \cdot \left(\frac{\mathcal{N}}{q\beta}\right)^N}. \qquad (11.15)$$

If k is the desired bit security level it will suffice to choose parameters so that the right hand side of (11.15) is less than 2^{-k}.

Signature Forgery Through Lattice Attacks

On the other hand, the adversary can also launch a lattice attack by attempting to solve a closest vector problem. In particular, he can attempt to use lattice reduction methods to locate a point $(s, \beta t) \in L_h(\beta)$ sufficiently close to $(0, \beta m)$ that $\|(s, \beta(t - m))\| < \mathcal{N}$. We will refer to $\|(s, \beta(t - m))\|$ as the norm of the intended forgery.

The difficulty of using lattice reduction methods to accomplish this can be tied to another important lattice constant:

$$\gamma(N, q, \beta) = \frac{\mathcal{N}}{\sigma(N, q, \delta, \beta)\sqrt{2N}}. \qquad (11.16)$$

Table 11.7 Bit security against lattice forgery attacks, ω_{lf}, based on experimental evidence for different values of $(\gamma, N/q)$

Bound for γ and N/q	$\omega_{\mathrm{lf}}(N)$
$\gamma < 0.1774$ and $N/q < 1.305$	$0.995113N - 82.6612$
$\gamma < 0.1413$ and $N/q < 0.707$	$1.16536N - 78.4659$
$\gamma < 0.1400$ and $N/q < 0.824$	$1.14133N - 76.9158$

This is the ratio of the required norm of the intended forgery over the norm of the expected smallest vector of $L_h(\beta)$, scaled by $\sqrt{2N}$. For usual NTRUSign parameters, the ratio, $\gamma(N, q, \beta)\sqrt{2N}$, will be larger than 1. Thus, with high probability, there will exist many points of $L_h(\beta)$ that will work as forgeries. The task of an adversary is to find one of these without the advantage that knowledge of the private key gives. As $\gamma(N, q, \beta)$ decreases and the ratio approaches 1, this becomes measurably harder.

Experiments have shown that for fixed $\gamma(N, q, \beta)$ and fixed N/q the running times for lattice reduction to find a point $(s, t) \in L_h(\beta)$ satisfying

$$\|(s, t - m)\| < \gamma(N, q, \beta)\sqrt{2N}\sigma(N, q, \delta, \beta)$$

behave roughly as

$$\log(T) = AN + B$$

as N increases. Here, A is fixed when $\gamma(N, q, \beta), N/q$ are fixed, increases as $\gamma(N, q, \beta)$ decreases and increases as N/q decreases. Experimental results are summarized in Table 11.7.

Our analysis shows that lattice strength against forgery is maximized, for a fixed N/q, when $\gamma(N, q, \beta)$ is as small as possible. We have

$$\gamma(N, q, \beta) = \rho\sqrt{\frac{\pi e}{2N^2 q} \cdot (\mathcal{E}_s^2/\beta + \beta\mathcal{E}_t^2)} \tag{11.17}$$

and so clearly the value for β which minimizes γ is $\beta = \mathcal{E}_s/\mathcal{E}_t$. This optimal choice yields

$$\gamma(N, q, \beta) = \rho\sqrt{\frac{\pi e \mathcal{E}_s \mathcal{E}_t}{N^2 q}}. \tag{11.18}$$

Referring to (11.15), we see that increasing β has the effect of improving combinatorial forgery security. Thus, the optimal choice will be the minimal $\beta \geq \mathcal{E}_s/\mathcal{E}_t$ such that $p(N, q, \beta, \mathcal{N})$ defined by (11.15) is sufficiently small.

An adversary could attempt a mixture of combinatorial and lattice techniques, fixing some coefficients and locating the others via lattice reduction. However, as explained in [17], the lattice dimension can only be reduced a small amount before a solution becomes very unlikely. Also, as the dimension is reduced, γ decreases, which sharply increases the lattice strength at a given dimension.

Transcript Security

NTRUSign is not zero-knowledge. This means that, while NTRUEncrypt can have provable security (in the sense of a reduction from an online attack method to a purely offline attack method), there is no known method for establishing such a reduction with NTRUSign. NTRUSign is different in this respect from established signature schemes such as ECDSA and RSA-PSS, which have reductions from online to offline attacks. Research is ongoing into quantifying what information is leaked from a transcript of signatures and how many signatures an attacker needs to observe to recover the private key or other information that would allow the creation of forgeries. This section summarizes existing knowledge about this information leakage.

Transcript Security for Raw NTRUSign

First, consider raw NTRUSign. In this case, an attacker studying a long transcript of valid signatures will have a list of pairs of polynomials of the form

$$\mathsf{s} = \epsilon \mathsf{f} + \epsilon' \mathsf{g}, \quad \mathsf{t} - \mathsf{m} = \epsilon \mathsf{F} + \epsilon' \mathsf{G}$$

where the coefficients of ϵ, ϵ' lie in the range $[-1/2, 1/2]$. In other words, the signatures lie inside a parallopiped whose sides are the good basis vectors. The attacker's challenge is to discover one edge of this parallelopiped.

Since the ϵs are random, they will average to 0. To base an attack on averaging s and $t-m$, the attacker must find something that does not average to zero. To do this, he uses the *reversal* of s and $t-m$. The reversal of a polynomial a is the polynomial

$$\bar{\mathsf{a}}(X) = \mathsf{a}(X^{-1}) = \mathsf{a}_0 + \sum_{i=1}^{N-1} \mathsf{a}_{N-i} X^i.$$

We then set

$$\hat{\mathsf{a}} = \mathsf{a} * \bar{\mathsf{a}}.$$

Notice that $\hat{\mathsf{a}}$ has the form

$$\hat{\mathsf{a}} = \sum_{k=0}^{N-1} \left(\sum_{i=0}^{N-1} \mathsf{a}_i \mathsf{a}_{i+k} \right) X^k.$$

In particular, $\hat{\mathsf{a}}_0 = \sum_i \mathsf{a}^2$. This means that as the attacker averages over a transcript of $\hat{\mathsf{s}}, \widehat{\mathsf{t}-\mathsf{m}}$, the cross-terms will essentially vanish and the attacker will recover

$$\langle \hat{\epsilon}_0 \rangle (\hat{\mathsf{f}} + \hat{\mathsf{g}}) = \frac{N}{12} (\hat{\mathsf{f}} + \hat{\mathsf{g}})$$

for s and similarly for $\mathsf{t} - \mathsf{m}$, where $\langle . \rangle$ denotes the average of . over the transcript.

We refer to the product of a measurable with its reverse as its *second moment*. In the case of raw NTRUSign, recovering the second moment of a transcript reveals the Gram Matrix of the private basis. Experimentally, it appears that significant information about the Gram Matrix is leaked after 10,000 signatures for all of the parameter sets in this paper. Nguyen and Regev [15] demonstrated an attack on parameter sets without perturbations that combines Gram matrix recovery with creative use of averaging moments over the signature transcript to recover the private key after seeing a transcript of approximately 70,000 signatures. This result has been improved to just 400 signatures in [24], and so the use of unperturbed NTRUSign is strongly discouraged.

Obviously, something must be done to reduce information leakage from transcripts, and this is the role played by perturbations.

Transcript Security for NTRUSign *with Perturbations*

In the case with B perturbations, the expectation of $\hat{\mathsf{s}}$ and $\hat{\mathsf{t}} - \hat{\mathsf{m}}$ is (up to lower order terms)

$$E(\hat{\mathsf{s}}) = (N/12)(\hat{\mathsf{f}}_0 + \hat{\mathsf{g}}_0 + \cdots + \hat{\mathsf{f}}_B + \hat{\mathsf{g}}_B)$$

and

$$E(\hat{\mathsf{t}} - \hat{\mathsf{m}}) = (N/12)(\hat{\mathsf{f}}_0 + \hat{\mathsf{g}}_0 + \cdots + \hat{\mathsf{f}}_B + \hat{\mathsf{g}}_B).$$

Note that this second moment is no longer a Gram matrix but the sum of $(B + 1)$ Gram matrices. Likewise, the signatures in a transcript do not lie within a parallelopiped but within the sum of $(B + 1)$ parallelopipeds.

This complicates matters for an attacker. The best currently known technique for $B = 1$ is to calculate

$$\begin{aligned} \text{the second moment} \quad & \langle \hat{\mathsf{s}} \rangle \\ \text{the fourth moment} \quad & \langle \hat{\mathsf{s}}^2 \rangle \\ \text{the sixth moment} \quad & \langle \hat{\mathsf{s}}^3 \rangle \ . \end{aligned}$$

Since, for example, $\langle \hat{\mathsf{s}} \rangle^2 \neq \langle \hat{\mathsf{s}}^2 \rangle$, the attacker can use linear algebra to eliminate f_1 and g_1 and recover the Gram matrix, whereupon the attack of [15] can be used to recover the private key. It is an interesting open research question to determine whether there is any method open to the attacker that enables them to eliminate the perturbation bases without recovering the sixth moment (or, in the case of B perturbation bases, the $(4B + 2)$-th moment). For now, the best known attack is this algebraic attack, which requires the recovery of the sixth moment. It is an open research problem to discover analytic attacks based on signature transcripts that improve on this algebraic attack.

We now turn to estimate τ, the length of transcript necessary to recover the sixth moment. Consider an attacker who attempts to recover the sixth moment by averaging over τ signatures and rounding to the nearest integer. This will give a reasonably correct answer when the error in many coefficients (say at least half) is less than $1/2$. To compute the probability that an individual coefficient has an error less than $1/2$, write $(12/N)\hat{s}$ as a main term plus an error, where the main term converges to $\hat{f}_0 + \hat{g}_0 + \hat{f}_1 + \hat{g}_1$. The error will converge to 0 at about the same rate as the main term converges to its expected value. If the probability that a given coefficient is further than $1/2$ from its expected value is less than $1/(2N)$, then we can expect at least half of the coefficients to round to their correct values (Note that this convergence cannot be speeded up using lattice reduction in, for example, the lattice $\hat{h}$, because the terms $\hat{f}$, $\hat{g}$ are unknown and are larger than the expected shortest vector in that lattice).

The rate of convergence of the error and its dependence on τ can be estimated by an application of Chernoff-Hoeffding techniques [48], using an assumption of a reasonable amount of independence and uniform distribution of random variables within the signature transcript. This assumption appears to be justified by experimental evidence and, in fact, benefits the attacker by ensuring that the cross-terms converge to zero.

Using this technique, we estimate that, to have a single coefficient in the $2k$-th moment with error less than $\frac{1}{2}$, the attacker must analyze a signature transcript of length $\tau > 2^{2k+4} d^{2k}/N$. Here, d is the number of 1s in the trinary key. Experimental evidence for the second moment indicates that the required transcript length will in fact be much longer than this. For one perturbation, the attacker needs to recover the sixth moment accurately, leading to required transcript lengths $\tau > 2^{30}$ for all the recommended parameter sets in this paper.

NTRUSign **Security: Summary**

The parameter sets in Table 11.8 were generated with $\rho = 1.1$ and selected to give the shortest possible signing time σ_S. These security estimates do *not* take the hybrid attack of [44] into account and are presented only to give a rough idea of the parameters required to obtain a given level of security.

The security measures have the following meanings:

ω_{lk}	The security against key recovery by lattice reduction
c	The lattice characteristic c that governs key recovery times
ω_{cmb}	The security against key recovery by combinatorial means
ω_{frg}	The security against forgery by combinatorial means
γ	The lattice characteristic γ that governs forgery times
ω_{lf}	The security against forgery by lattice reduction

Table 11.8 Parameters and relevant security measures for trinary keys, one perturbation, $\rho = 1.1$, q = power of 2

Parameters						Security measures						
k	N	d	q	β	$\mathcal{N}$	ω_{cmb}	c	ω_{lk}	ω_{frg}	γ	ω_{lf}	$\log_2(\tau)$
80	157	29	256	0.38407	150.02	104.43	5.34	93.319	80	0.139	102.27	31.9
112	197	28	256	0.51492	206.91	112.71	5.55	117.71	112	0.142	113.38	31.2
128	223	32	256	0.65515	277.52	128.63	6.11	134.5	128	0.164	139.25	32.2
160	263	45	512	0.31583	276.53	169.2	5.33	161.31	160	0.108	228.02	34.9
192	313	50	512	0.40600	384.41	193.87	5.86	193.22	192	0.119	280.32	35.6
256	349	75	512	0.18543	368.62	256.48	7.37	426.19	744	0.125	328.24	38.9

Quantum Computers

All cryptographic systems based on the problems of integer factorization, discrete log, and elliptic curve discrete log are potentially vulnerable to the development of an appropriately sized quantum computer, as algorithms for such a computer are known that can solve these problems in time polynomial in the size of the inputs. At the moment, it is unclear what effect quantum computers may have on the security of the NTRU algorithms.

The paper [28] describes a quantum algorithm that square-roots asymptotic lattice reduction running times for a specific lattice reduction algorithm. However, since, in practice, lattice reduction algorithms perform much better than they are theoretically predicted to, it is not clear what effect this improvement in asymptotic running times has on practical security. On the combinatorial side, Grover's algorithm [49] provides a means for square-rooting the time for a brute-force search. However, the combinatorial security of NTRU keys depends on a meet-in-the-middle attack, and we are not currently aware of any quantum algorithms to speed this up. The papers [50–54] consider potential sub-exponential algorithms for certain lattice problems. However, these algorithms depend on a subexponential number of coset samples to obtain a polynomial approximation to the shortest vector, and no method is currently known to produce a subexponential number of samples in subexponential time.

At the moment, it seems reasonable to speculate that quantum algorithms will be discovered that will square-root times for both lattice reduction and meet-in-the-middle searches. If this is the case, NTRU key sizes will have to approximately double, and running times will increase by a factor of approximately 4 to give the same security levels. As demonstrated in the performance tables in this paper, this still results in performance that is competitive with public key algorithms that are in use today. As quantum computers are seen to become more and more feasible, NTRUEncrypt and NTRUSign should be seriously studied with a view to wide deployment.

References

1. M. Ajtai, *The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract)*, in Proc. 30th ACM symp on Theory of Computing, pp. 10–19, 1998
2. O. Goldreich, D. Micciancio, S. Safra, J.-P. Seifert, *Approximating shortest lattice vectors is not harder than approximating closest lattice vectors*, in Inform. Process. Lett. 71(2), 55–61, 1999
3. D. Micciancio, *Complexity of Lattice Problems*, Kluwer International Series in Engineering and Computer Science, vol. 671 Kluwer, Dordrecht, March 2002
4. H. Cohn, A. Kumar, *The densest lattice in twenty-four dimensions* in Electron. Res. Announc. Amer. Math. Soc. 10, 58–67, 2004
5. R.C. Merkle, M.E. Hellman, *Hiding information and signatures in trapdoor knapsacks*, in Secure communications and asymmetric cryptosystems, AAAS Sel. Sympos. Ser, 69, 197–215, 1982
6. A.M. Odlyzko, *The rise and fall of knapsack cryptosystems*, in Cryptology and computational number theory (Boulder, CO, 1989), Proc. Sympos. Appl. Math. 42, 75–88, 1990
7. A.K. Lenstra, A.K., H.W. Lenstra, L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann. 261, 515–534, 1982
8. M. Ajtai, C. Dwork, *A public-key cryptosystem with worst- case/average-case equivalence*, in Proc. 29th Annual ACM Symposium on Theory of Computing (STOC), pp. 284–293, ACM Press, New York, 1997
9. O. Goldreich, S. Goldwasser, S. Halevi, *Public-key cryptosystems from lattice reduction problems, advances in cryptology*, in Proc. Crypto 97, Lecture Notes in Computer Science, vol. 1294, pp. 112–131, Springer, Berlin, 1997
10. J. Hoffstein, J. Pipher, J.H. Silverman, *NTRU: A new high speed public key cryptosystem*, in J.P. Buhler (Ed.), Algorithmic Number Theory (ANTS III), Portland, OR, June 1998, Lecture Notes in Computer Science 1423, pp. 267–288, Springer, Berlin, 1998
11. P. Nguyen, J. Stern, *Cryptanalysis of the Ajtai-Dwork cryptosystem*, in Proc. of Crypto '98, vol. 1462 of LNCS, pp. 223–242, Springer, Berlin, 1998
12. L. Babai, *On Lovasz Lattice Reduction and the Nearest Lattice Point Prob- lem*, Combinatorica, vol. 6, pp. 113, 1986
13. P. Nguyen, *Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto '97*, in Crypto'99, LNCS 1666, pp. 288–304, Springer, Berlin, 1999
14. J. Hoffstein, J.H. Silverman, W. Whyte, *Estimated Breaking Times for NTRU Lattices*, Technical report, NTRU Cryptosystems, June 2003 Report #012, version 2, Available at http://www.ntru.com
15. P. Nguyen, O. Regev, *Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures*, Eurocrypt, pp. 271–288, 2006
16. J. Hoffstein, J. Pipher, J.H. Silverman, *NSS: The NTRU signature scheme*, in B. Pfitzmann (Ed.), Eurocrypt '01, Lecture Notes in Computer Science 2045, pp. 211–228, Springer, Berlin, 2001
17. J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. Silverman, W. Whyte, *NTRUSign: Digital Signatures Using the NTRU Lattice*, CT-RSA, 2003
18. J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. Silverman, W. Whyte, *NTRUSign: Digital Signatures Using the NTRU Lattice, extended version*, Available from http://ntru.com/cryptolab/pdf/NTRUSign-preV2.pdf
19. N. Howgrave-Graham, P. Nguyen, D. Pointcheval, J. Proos, J.H. Silverman, A. Singer, W. Whyte, *The Impact of Decryption Failures on the Security of NTRU Encryption*, Advances in Cryptology – Crypto 2003, Lecture Notes in Computer Science 2729, pp. 226–246, Springer, Berlin, 2003
20. J. Proos, *Imperfect Decryption and an Attack on the NTRU Encryption Scheme*, IACR ePrint Archive, report 02/2003, Available at http://eprint.iacr.org/2003/002/
21. Consortium for Efficient Embedded Security, *Efficient Embedded Security Standard #1 version 2*, Available from http://www.ceesstandards.org

22. C. Gentry, J. Jonsson, J. Stern, M. Szydlo, *Cryptanalysis of the NTRU signature scheme, (NSS), from Eurocrypt 2001*, in Proc. of Asiacrypt 2001, Lecture Notes in Computer Science, pp. 1–20, Springer, Berlin, 2001
23. C. Gentry, M Szydlo, *Cryptanalysis of the Revised NTRU SignatureScheme*, Advances in Cryptology – Eurocrypt '02, Lecture Notes in Computer Science, Springer, Berlin, 2002
24. P.Q. Nguyen, *A Note on the Security of NTRUSign*, Cryptology ePrint Archive: Report 2006/387
25. N. Howgrave-Graham, J.H. Silverman, W. Whyte, *Choosing Parameter Sets for* NTRUEncrypt *with* NAEP *and* SVES-3, CT-RSA, 2005
26. J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. Silverman, W. Whyte, *Performance Improvements and a Baseline Parameter Generation Algorithm for* NTRUSign, Workshop on Mathematical Problems and Techniques in Cryptology, Barcelona, Spain, June 2005
27. P. Shor, *Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer*, Preliminary version appeared in Proc. of 35th Annual Symp. on Foundations of Computer Science, Santa Fe, NM, Nov 20–22, 1994. Final version published in SIAM J. Computing 26 (1997) 1484, Published in SIAM J. Sci. Statist. Comput. 26, 1484, 1997, e-Print Archive: quant-ph/9508027
28. C. Ludwig, *A Faster Lattice Reduction Method Using Quantum Search*, TU-Darmstadt Cryptography and Computeralgebra Technical Report No. TI-3/03, revised version published in Proc. of ISAAC 2003
29. J. Hoffstein, J.H. Silverman, *Invertibility in truncated polynomial rings*, Technical report, NTRU Cryptosystems, October 1998, Report #009, version 1, Available at http://www.ntru.com
30. N. Howgrave-Graham, J.H. Silverman, A. Singer, W. Whyte, *NAEP: Provable Security in the Presence of Decryption Failures*, IACR ePrint Archive, Report 2003-172, http://eprint.iacr.org/2003/172/
31. M. Bellare, P. Rogaway, *Optimal asymmetric encryption*, in Proc. of Eurocrypt '94, vol. 950 of LNCS, IACR, pp. 92–111, Springer, Berlin, 1995
32. D. Boneh, *Simplified OAEP for the RSA and Rabin functions*, in Proc. of Crypto '2001, Lecture Notes in Computer Science, vol. 2139, pp. 275–291, Springer, Berlin, 2001
33. M. Brown, D. Hankerson, J. López, A. Menezes, *Software Implementation of the NIST Elliptic Curves Over Prime Fields* in D. Naccache (Ed.), CT-RSA 2001, LNCS 2020, pp. 250–265, Springer, Berlin, 2001
34. A.K. Lenstra, E.R. Verheul, *Selecting cryptographic key sizes*, J. Cryptol. 14(4), 255–293, 2001, Available from http://www.cryptosavvy.com
35. R.D. Silverman, *A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths*, RSA Labs Bulletin 13, April 2000, Available from http://www.rsasecurity.com/rsalabs
36. NIST Special Publication 800-57, *Recommendation for Key Management, Part 1: General Guideline*, January 2003, Available from http://csrc.nist.gov/CryptoToolkit/kms/guideline-1-Jan03.pdf
37. B. Kaliski, *Comments on SP 800-57, Recommendation for Key Management, Part 1: General Guidelines*, Available from http://csrc.nist.gov/CryptoToolkit/kms/CommentsSP800-57Part1.pdf
38. R. Rivest, A. Shamir, L.M. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Commun. ACM 21, 120–126, 1978
39. N. Koblitz, *Elliptic curve cryptosystems*, Mathematics of Computation, 48, pp. 203–209, 1987
40. V. Miller, *Uses of elliptic curves in cryptography*, in Advances in Cryptology: Crypto '85, pp. 417–426, 1985
41. N. Howgrave-Graham, J.H. Silverman, W. Whyte, *A Meet-in-the-Middle Attack on an NTRU Private key*, Technical report, NTRU Cryptosystems, June 2003, Report #004, version 2, Available at `http://www.ntru.com`
42. D. Coppersmith, A. Shamir, *Lattice Attack on NTRU*, Advances in Cryptology – Eurocrypt 97, Springer, Berlin

AQ1

43. A. May, J.H. Silverman, *Dimension reduction methods for convolution modular lattices*, in J.H. Silverman (Ed.), Cryptography and Lattices Conference (CaLC 2001), Lecture Notes in Computer Science 2146, Springer, Berlin, 2001
44. N. Howgrave-Graham, *A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU*, Lecture Notes in Computer Science, Springer, Berlin, in Advances in Cryptology – CRYPTO 2007, vol. 4622/2007, pp. 150–169, 2007
45. P. Hirschhorn, J. Hoffstein, N. Howgrave-Graham, W. Whyte, *Choosing NTRU Parameters in Light of Combined Lattice Reduction and MITM Approaches* AQ2
46. C. Gentry, Key Recovery and Message Attacks on NTRU-Composite, *Advances in Cryptology – Eurocrypt '01*, LNCS 2045, Springer, Berlin, 2001
47. J.H. Silverman, *Invertibility in Truncated Polynomial Rings*, Technical report, NTRU Cryptosystems, October 1998, Report #009, version 1, Available at http://www.ntru.com
48. Kirill Levchenko, *Chernoff Bound*, Available at http://www.cs.ucsd.edu/~klevchen/techniques/chernoff.pdf
49. L. Grover, *A fast quantum mechanical algorithm for database search*, in Proc. 28th Annual ACM Symposium on the Theory of Computing, 1996
50. O. Regev, *Quantum computation and lattice problems*, in Proc. 43rd Annual Symposium on the Foundations of Computer Science, pp. 520–530, IEEE Computer Society Press, Los Alamitos, California, USA, 2002, http://citeseer.ist.psu.edu/regev03quantum.html
51. T. Tatsuie, K. Hiroaki, *Efficient algorithm for the unique shortest lattice vector problem using quantum oracle*, IEIC Technical Report, Institute of Electronics, Information and Communication Engineers, vol. 101, No. 44(COMP2001 5–12), pp. 9–16, 2001
52. Greg Kuperberg, *A Sub-Exponential-Time Quantum Algorithm For The Dihedral Hidden Subgroup Problem*, 2003, http://arxiv.org/abs/quant-ph/0302112
53. O. Regev, *A Sub-Exponential Time Algorithm for the Dihedral Hidden Subgroup Problem with Polynomial Space*, June 2004, http://arxiv.org/abs/quant-ph/0406151
54. R. Hughes, G. Doolen, D. Awschalom, C. Caves, M. Chapman, R. Clark, D. Cory, D. DiVincenzo, A. Ekert, P. Chris Hammel, P. Kwiat, S. Lloyd, G. Milburn, T. Orlando, D. Steel, U. Vazirani, B. Whaley, D. Wineland, *A Quantum Information Science and Technology Roadmap, Part 1: Quantum Computation*, Report of the Quantum Information Science and Technology Experts Panel, Version 2.0, April 2, 2004, Advanced Research and Development Activity, http://qist.lanl.gov/pdfs/qc_roadmap.pdf AQ3
55. ANSI X9.62, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), 1999
56. D. Hankerson, J. Hernandez, A. Menezes, *Software implementation of elliptic curve cryptography over binary fields*, in Proc. CHES 2000, Lecture Notes in Computer Science, 1965, pp. 1–24, 2000
57. J. Hoffstein, J.H. Silverman, *Optimizations for NTRU*, In Publickey Cryptography and Computational Number Theory. DeGruyter, 2000, Available from http://www.ntru.com
58. J. Hoffstein, J.H. Silverman, *Random Small Hamming Weight Products with Applications to Cryptography*, Discrete Applied Mathematics, Available from http://www.ntru.com
59. E. Kiltz, J. Malone-Lee, *A General Construction of IND-CCA2 Secure Public Key Encryption*, in Cryptography and Coding, pp. 152–166, Springer, Berlin, December 2003
60. T. Meskanen, A. Renvall, *Wrap Error Attack Against NTRUEncrypt*, in Proc. of WCC '03, 2003
61. NIST, *Digital Signature Standard*, FIPS Publication 186-2, February 2000

AUTHOR QUERIES

AQ1. Au: Kindly provide complete details for the reference [42].
AQ2. Au: Please update the reference [45].
AQ3. Au: Please provide citation for the references [55]–[61]