A signature scheme from Learning with Truncation

Jeffrey Hoffstein[1], Jill Pipher[1], William Whyte[2], and Zhenfei Zhang[2]

[1] Brown University, Providence RI, USA, {jhoff,jpipher}@math.brown.edu
[2] OnBoard Security, Wilmington MA, USA, {wwhyte,zzhang}@onboardsecurity.com

Abstract. In this paper we revisit the modular lattice signature scheme and its efficient instantiation known as pqNTRUSign. First, we show that a modular lattice signature scheme can be based on a standard lattice problem. As the fundamental problem that needs to be solved by the signer or a potential forger is recovering a lattice vector with a restricted norm, given the least significant bits, we refer to this general class of problems as the "learning with truncation" problem.
We show that by replacing the uniform sampling in pqNTRUSign with a bimodal Gaussian sampling, we can further reduce the size of a signature. As an example, we show that the size of the signature can be as low as 4608 bits for a security level of 128 bits.
The most significant new contribution, enabled by this Gaussian sampling version of pqNTRUSign, is that we can now perform batch verification, which allows the verifier to check approximately 2000 signatures in a single verification process.

1 Introduction

Organizations and research groups are looking for candidate algorithms to replace RSA and ECC based schemes [7, 1] due to the threat of quantum computers [30]. Among all candidates, lattice based solutions seem to offer the most promising solutions. For encryption schemes, NTRUEncrypt [19] is known to be one of the fastest lattice based encryption algorithms to date. In terms of signature schemes, there exist several lattice based schemes that rely on the security of the NTRU lattice [19], such as BLISS [9], DLP [11] and pqNTRUSign [17].

In this paper, we revisit the modular lattice signature scheme from [17]. Given a lattice $\mathcal{L}$ with a trapdoor $\mathbf{T}$ in the form of a short basis of row vectors; and a message digest in the form of a vector $\mathbf{m}$ whose coefficients are in $[0, p)$, the signature of a modular signature scheme in [17] is a lattice vector $\mathbf{v}$ such that

1. $\mathbf{v} \equiv \mathbf{m} \bmod p$; and
2. $\mathbf{v} \in \mathcal{L}$.

This vector can be obtained via the following steps:

1. sample a vector $\mathbf{v}_0$ from $\mathcal{L}$;

2. use $\mathbf{T}$ to micro-adjust the $\mathbf{v}_0$ so that $\mathbf{v} := \mathbf{v}_0 + \mathbf{aT}$ meets the congruence condition for some $\mathbf{a}$;
3. use rejection sampling to hide $\mathbf{T}$ from $\mathbf{v}$

In this paper we revisit the above scheme from the following perspectives.

Security proof. In the original modular lattice signature scheme in [17], the public key security is connected to a unique shortest non-zero vector problem (uSVP), i.e., recovering $\mathbf{T}$ from (a bad basis of) $\mathcal{L}$; while the unforgeability is based on an approximate closest vector problem (approx-CVP) over the intersection of a lattice and a translation of a lattice, namely, $\mathcal{L} \cap (p\mathbb{Z}^n + \mathbf{m}_p)$. Although the second problem is conjectured to be hard for this ad hoc lattice, a connection between the first and the second problems is missing. The scheme, therefore, requires two (seemingly independent) hardness assumptions. For example, when the scheme is instantiated via an NTRU lattice [17], they require the uSVP assumption for the NTRU lattice and the above approx-CVP assumption for the intersection of the two lattices.

In this paper, we remove the second assumption. Essentially, the attacker is given a lattice (any lattice, not necessarily an NTRU lattice), and he is asked to find a vector in the lattice such that this vector is congruent to a known value mod p. In other words, the attacker needs to find a vector with pre-determined least significant bits. We name this problem the *learning with truncation* (LWT) problem, which can be viewed as the inverse of learning with rounding (LWR) problem, where in this case one is given matrix $\mathbf{A}$ and vector $\mathbf{b} = \lfloor \mathbf{sA} \bmod q \rceil_p$, and asked to find $\mathbf{s}$. That is, to find a vector related to the lattice where the most significant bits are pre-determined.

This allows us to connect the forgery attack with the approx-SVP. As a result, we only require a single assumption. In particular, when the scheme is instantiated via a Short Integer Solution (SIS) problem, forging a signature in our scheme is as hard as solving an approx-SVP for random lattices. On the other hand, when the scheme is instantiated via an NTRU lattice, we require that approx-SVP is hard for NTRU lattices, which is equivalent to the unique-SVP assumption (up to a polynomial factor, cf. [24]), a.k.a., the NTRU assumption .

Sampling method. Early lattice based signature schemes, such as GGHSign [15] and NTRUSign [16], leak private key information in a transcript of message/signature pairs. An attacker can produce a signing key from a long enough transcript using methods for "learning a parallelepiped" [27, 12].

In [21], Lyubashevsky proposed a rejection sampling method to thwart transcript leakage attacks. Using his technique, signatures are produced according to a fixed public distribution (typically either a Gaussian as in [21] or a uniform distribution as in [17]). A transcript reveals only this public distribution, and contains no information about the particular signing key that is used to generate the signatures. The sampling method therefore becomes a core issue in designing signature schemes. For example, replacing a Gaussian sampler with a bimodal Gaussian sampler [9] significantly improves the performance of a scheme.

Recall that in [17], a signature in this scheme is a lattice vector. Since the verifier already knows a (bad) basis of the lattice for verification purpose, it is sufficient to transmit part of the vector $\mathbf{v}$ as long as the verifier can complete the whole vector during the verification phase.

Popular lattice based schemes, such as BLISS [9] and TESLA [3], do not have this property. Signatures in those schemes are vectors *close* to the lattice. Hence, when the vectors are compressed, an additional helper needs to be generated for the verifier to derive the original vector (although this helper is only a few hundred bits). To be precise, if we parameterize the scheme to be presented in this paper with the same parameters as in [9], the difference in the size of a signature is exactly the size of this helper.

This advantage in design did not give a smaller signature size for [17] due to the sampling method. For an n-dimensional vector with coefficients in $[-\frac{q}{2}, \frac{q}{2})$, it requires $n\lceil \log q \rceil$ bits for storage. For comparison, a discrete Gaussian vector of the same dimension with a deviation of $\sigma \sim \sqrt{q}$ can be stored with $\sim n(\frac{\log q}{2} + 2)$ bits. A natural question is whether one can use (bimodal) Gaussian sampling [9] for modular lattice signatures. In this paper, we give a positive answer to this question.

Remark 1. Although schemes using Gaussian sampling allow smaller signature sizes, recent development in lattice based signature schemes [10] shows a trend of moving back to uniform rejection sampling since uniform sampling is easier to implement and to ensure constant time. Nevertheless, with pqNTRUSign, Gaussian sampling enables us to obtain an additional property: signature aggregation.

Signature aggregation. Signature aggregation, also known as batch verification, allows one to verify a set of signatures, signed under a same key, with operations that are on the order of a single verification. It is a useful property in many use cases. As an example, for a secure boot mechanism where the software image is signed, signature aggregation allows one to sign individual software images individually (and do so component wise rather than monolithic updates) while still verifying the entire set of software images in one pass. This allows for fast boot.

Our scheme allows for batch verification (with fine-tuned parameters). Generally speaking, a signature $\mathbf{v}$ for a message digest $\mathbf{m}$ is valid so long as $\mathbf{v} \equiv \mathbf{m} \bmod p$ and $\mathbf{v} \in \mathcal{L}$. Therefore, for a set of signatures $\{\mathbf{v}_i\}$, corresponding to a set of messages $\{\mathbf{m}_i\}$ we have

1. $\sum \mathbf{v}_i \equiv \sum \mathbf{m}_i \bmod p$; and
2. $\sum \mathbf{v}_i \in \mathcal{L}$.

As such, one can simply check $\sum \mathbf{v}_i$ instead of checking each individual $\mathbf{v}$. When realizing this technique for our proposed scheme, we can use a single ring multiplication (which is usually the most costly operation in verification) to verify a batch of signatures. Nevertheless we note that one will still need to perform multiple hash functions to obtain those message digests. In addition, since the

accumulated signature is a larger vector in the lattice (compared to a single signature), we will require that the corresponding lattice problem for this accumulated signature is also hard. We will give more details in section 5.

We also note that other lattice-based schemes such as BLISS [9] and TESLA [3], cannot provide this property easily as they need to perform the ring operations before the hash function.

Paper Organization. In section 2 we give some background to this work. In section 3 we give a modular lattice signature scheme based on the short integer solution problem. We show that to forge a signature is as hard as solving the computational LWR problem for a random lattice. This is followed by a practical instantiation using NTRU lattices and a bimodal Gaussian in section 4. Then we explain signature aggregation in more details in section 5 and present parameters for our practical instantiation in section 6.

2 Background

2.1 Notations

We use lower case bold letters for vectors, upper case bold letters for matrices. For a polynomial $f(x) = f_0 + f_1 x + \cdots + f_{n-1}x^{n-1}$, we denote its vector form by $\mathbf{f} \coloneqq \langle f_0, f_1, \ldots, f_{n-1}\rangle$. We sometimes abuse the notation of vector and polynomial when there is no ambiguity. For a polynomial/vector $\mathbf{f}$, the norms are $\|\mathbf{f}\| \coloneqq \sqrt{\sum_{i=0}^{n-1} f_i^2}$ and $\|\mathbf{f}\|_\infty \coloneqq \max(|f_i|)$.

We often use the polynomial rings $\mathcal{R}_q \coloneqq \mathbb{Z}[x]/F(x)$ with $F(x) = x^n \pm 1$. A cyclic rotated matrix of a polynomial $f(x)$ over the ring $\mathcal{R}_q$ is a matrix $\mathbf{M} = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n)^T$ with $\mathbf{f}_i = f(x)x^{i-1} \bmod F(x)$. If $F(x) = x^n - 1$ it is literally cyclic, and close to cyclic, up to signs, if $F(x) = x^n + 1$.

For a real a, we let $\lfloor a \rceil$ denote the closet integer to a. For an integer a, we use $[a]_q$ to denote $a \bmod q$; $\lfloor a \rfloor_p \coloneqq (a - [a]_p)/p$ for the operation of rounding a to the closest multiple of p. Modular operations are center lifted, for example $a \bmod q$ returns an integer within $-q/2$ and $q/2$. These notations are also extended to vectors and matrices.

2.2 NTRU, SIS, LWR and lattices

A lattice $\mathcal{L}$ is a discrete sub-group of $\mathbb{R}^n$, or equivalently, the set of all the integral combinations of $d \leq n$ linearly independent vectors over $\mathbb{R}$:

$$\mathcal{L} \coloneqq \mathbb{Z}\mathbf{b}_1 + \mathbb{Z}\mathbf{b}_2 + \cdots + \mathbb{Z}\mathbf{b}_d, \mathbf{b}_i \in \mathbb{R}^n.$$

$\mathbf{B} \coloneqq (\mathbf{b}_1, \ldots, \mathbf{b}_d)^T$ is called a basis of $\mathcal{L}$. Given a lattice $\mathcal{L}$, finding a vector that is no longer than $\gamma \cdot \lambda_1(\mathcal{L})$ is called the approximate shortest vector problem (γ-SVP), where λ_1 is the first minima, i.e, the length of the shortest vector, of the lattice. The Gaussian heuristic says that for random lattices, this first

minima should be approximately $\lambda_1 \approx \sqrt{\frac{\dim}{2\pi e}} \det(\mathcal{L})^{\frac{1}{\dim}}$, where $\det(\mathcal{L})$ denotes the determinant of $\mathcal{L}$. Given a particular lattice $\mathcal{L}$, where there exists a unique shortest non-zero vector, finding this vector is called the unique shortest vector problem.

We view an NTRU lattice as an $\mathcal{R}_q$ module of rank 2. Let $\mathbf{f}, \mathbf{g} \in \mathcal{R}_q$ with small coefficients. Let $\mathbf{h} = \mathbf{g}/\mathbf{f}$ over $\mathcal{R}_q$. The NTRU lattice associated with $\mathbf{h}$ is defined as

$$\mathcal{L} \coloneqq \{(\mathbf{s}, \mathbf{t}) \in \mathcal{R}_q^2 : \mathbf{t} \equiv \mathbf{sh} \mod q\}.$$

Given $\mathbf{h}$, it is believed to be hard to find $\mathbf{f}$ and $\mathbf{g}$. This is known as the *NTRU assumption*, and it can be reduced to the unique shortest vector problem for the NTRU lattice.

We write a vector in the NTRU lattice as $\mathbf{v} = \langle \mathbf{s}, \mathbf{t} \rangle$, where $\mathbf{s}$ and $\mathbf{t}$ are each an element in $\mathcal{R}_q$. In addition, we refer to the sub-vector that forms the first part of this vector as the "s-side" vector, and that which forms the second part of this vector as the "t-side" vector.

We extend this notion to the short integer solution problem (SIS) when applicable. Recall that an SIS problem is defined as follows:

Definition 1 ($\mathbf{SIS}_{q,n,m,\beta}$ problem). *Given a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find a short non-zero vector $\mathbf{v}$ such that $\mathbf{vA} \equiv 0 \bmod q$ with $\|\mathbf{v}\|_2 \leq \beta$.*

For a matrix $\mathbf{A}$ that is a horizontal concatenation of two matrices, i.e., $\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix}$, the lattice associated with $\mathbf{A}$ is defined as

$$\mathcal{L} \coloneqq \{(\mathbf{s}, \mathbf{t}) : \mathbf{sA}_1 + \mathbf{tA}_2 \equiv 0 \bmod q\}.$$

Finding a short $(\mathbf{s}, \mathbf{t})$ in this lattice provides a solution to the SIS problem. It was shown in [25] that solving SIS on average for $n = \text{poly}(m)$, $q \geq \beta \cdot m^\delta$ for some positive δ, is as hard as the shortest independent vector problem with approximating factor $\max\{1, \beta\beta_\infty/q\} \cdot O(\beta\sqrt{m})$ where β_∞ is the upper bound for the infinity norm of $\mathbf{v}$.

The SIS problem has a "dual" version, known as the LWE problem. Informally speaking, let m, n, q be some positive integers, let χ_σ be an error distribution parameterized by σ, for example, a discrete Gaussian distribution with standard deviation σ, sample uniformly at random $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{s}, \mathbf{b}_1 \in \mathbb{Z}_q^n$; sample $\mathbf{e} \in \chi_\sigma^m$; compute $\mathbf{b}_0 = \mathbf{sA} + \mathbf{e} \bmod q$; the decisional LWE assumption states that given two pairs $(\mathbf{A}, \mathbf{b}_0)$, with $\mathbf{b}_0$ generated as above; and $(\mathbf{A}, \mathbf{b}_1)$, with , $\mathbf{b}_1$ chosen from a uniform distribution, one is not able to distinguish those two pairs.

We also make use of the learning with rounding (LWR) problem [6, 5]. This can be seen as a variant of the learning with errors (LWE) problem [29], with deterministic errors from rounding. We formally record the LWR problem as follows:

Definition 2 ($\mathbf{LWR}_{q,r,n,m}$ problem). *Sample uniformly at random a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a vector $\mathbf{s} \in \mathbb{Z}_q^n$; compute $\mathbf{b} = \lfloor \mathbf{sA} \bmod q \rfloor_r$; the decisional LWR problem is: given two pairs $(\mathbf{A}, \mathbf{b})$ and $(\mathbf{A}, \lfloor \mathbf{u} \rfloor_r)$ where $\mathbf{u}$ is sampled uniformly at*

random in $\mathbb{Z}_q^n$*, distinguish those two pairs. The computational problem is: given* $(\mathbf{A}, \mathbf{b})$*, find* $\mathbf{s}$*.*

It has been shown in [5] that the decisional $\mathrm{LWR}_{q,r,n,m}$ problem is hard assuming the hardness of $\mathrm{LWE}_{q,r,n,m'}$ with parameters

$$m \geq \frac{\log(q)}{\log(2\gamma)} \cdot m' \qquad \text{and} \qquad q \geq \gamma(nm\beta p)$$

for some $\gamma \geq 1$. To the best of our knowledge, we are not aware of reductions between computational LWR and other assumptions.

2.3 Bimodal Gaussian distribution and rejection sampling

An n-dimensional Gaussian distribution with mean $\mathbf{v}$ and standard deviation σ is defined by $\rho_{\mathbf{v},\sigma}(\mathbf{x}) \coloneqq \exp(\frac{-\|\mathbf{x}-\mathbf{v}\|^2}{2\sigma^2})$. When there is no ambiguity, we abbreviate this by ρ_σ. An n-dimensional discrete Gaussian distribution over $\mathbb{Z}$ is defined by $\chi_\sigma \coloneqq \frac{\rho_\sigma(\mathbf{x})}{\rho_\sigma(\mathbb{Z}^n)}$, where $\rho_\sigma(\mathbb{Z}^n) \coloneqq \sum_{\mathbf{z}\in\mathbb{Z}^n} \rho_\sigma(\mathbf{z})$ is a scaling quantity needed to make the function into a probability distribution [22].

Tail cutting: For a discrete Gaussian distribution χ_σ^m and a positive $\tau > 1$,

$$\rho_\sigma(\mathbb{Z}^m \backslash \tau\sigma\sqrt{m}\mathcal{B}) \leq 2\rho_\sigma(\mathbb{Z}^m)\left(\tau \exp\left(\frac{1-\tau^2}{2}\right)\right)^m,$$

where $\mathcal{B}$ is the centered unit ball [26]. As suggested in [9], setting $\tau = \sqrt{\lambda 2 \ln 2}$ for a 1-dimensional Gaussian will ensure all samples are bounded by $\tau\sigma$ with a probability greater than $1 - 2^{-\lambda}$. Typically, $\tau = 13.3$ for $\lambda = 128$ and $\tau = 18.8$ for $\lambda = 256$.

Rejection sampling: Let $\mathbf{S}$ be a secret matrix, $\mathbf{c}$ be a vector sampled from a uniform distribution, and $\mathbf{y}$ be a vector sampled from χ_σ. Consider the distribution of $\mathbf{x} = \mathbf{y} + \mathbf{cS}$, i.e., a Gaussian distribution shifted by $\mathbf{cS}$. It has been shown in [27, 12] that each sample $\mathbf{x}$ leaks partial information on $\mathbf{S}$. The method used to seal this leakage is rejection sampling [21]: making the output distribution independent of $\mathbf{S}$ by probabilistically accepting the output according to certain criteria.

As shown in [23], if we wish to force output distribution to be the same as $\mathbf{y}$, it is sufficient to have

$$\frac{\chi_\sigma(\mathbf{x})}{\chi_{\mathbf{cS},\sigma}(\mathbf{x})} \leq M,$$

and this inequality holds with

$$M = \exp\left(\frac{2\tau\sigma \max_{\mathbf{c}} \|\mathbf{cS}\| + \max_{\mathbf{c}} \|\mathbf{cS}\|^2}{2\sigma^2}\right)$$

where M is the repetition rate. The constant M determines the rate of rejection, and the smaller M is, the more efficient the signature generation process is. A common choice is to set $\sigma = \tau \max_{\mathbf{c}} \|\mathbf{cS}\|$ which gives a constant (while still largish) M. This is improved when bimodal Gaussian sampling is used [9].

Bimodal Gaussian: Informally speaking, a bimodal Gaussian is a sum of two Gaussian distributions with the same σ and means of the same absolute value, with opposite signs. Following the above example, the distribution of $\mathbf{x} = \mathbf{y} \pm \mathbf{cS}$ is very close to a bimodal Gaussian distribution. One can use rejection sampling to produce the Gaussian distribution χ_σ from the bimodal Gaussian distribution $\frac{1}{2}\chi_{\mathbf{cS},\sigma}(\mathbf{x}) + \frac{1}{2}\chi_{-\mathbf{cS},\sigma}(\mathbf{x})$ if there exists a constant M such that

$$\frac{\chi_\sigma(\mathbf{x})}{\frac{1}{2}\chi_{\mathbf{cS},\sigma}(\mathbf{x}) + \frac{1}{2}\chi_{-\mathbf{cS},\sigma}(\mathbf{x})} \leq M.$$

It has been shown in [9] that this inequality holds with

$$M = \exp\left(\frac{\max_{\mathbf{c}}(\|\mathbf{cS}\|^2)}{2\sigma^2}\right). \tag{1}$$

It is also shown in [9], that for an individual $\mathbf{x} = \mathbf{y} \pm \mathbf{cS}$, the probability of accepting it is given by

$$\rho = 1/\left(M \exp\left(-\frac{\|\mathbf{cS}\|}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{x}, \mathbf{cS}\rangle}{\sigma^2}\right)\right). \tag{2}$$

Remark 2. As usual there is a trade-off between efficiency and storage size. For the discrete Gaussian distribution χ_σ, the entropy of its output $\mathbf{x}$ is bounded above by

$$\mathcal{H}(\mathbf{x}) \lessapprox k \log(4.1\sigma).$$

Therefore, such a vector can be efficiently stored with approximately $k(\log(\sigma)+2)$ bits, using Hoffman coding. Thus a smaller σ yields a smaller signature, but simultaneously makes rejection sampling less efficient.

3 Modular lattice signatures with Gaussian sampling

3.1 The scheme

Construction: Let m, n and k be 3 positive integers with $n = k + m$. Let $\mathbf{S}_1 \in \mathbb{Z}_q^{m \times k}$ be a matrix with small (and sparse) coefficients. For simplicity, we assume $\mathbf{S}_1$ is sampled from a certain β-bounded sampler such that $\|\mathbf{S}_1\|_\infty \leq \beta \ll q$. In practice one can use either a discrete Gaussian sampler with small variance, or a uniform sampler within a small range.

Our secret key is a matrix $\mathbf{S} := [p\mathbf{S}_1|\mathbf{I}_m] \in \mathbb{Z}_q^{m \times n}$ with small entries. The public key is constructed from a matrix $\mathbf{A} = \begin{bmatrix}\mathbf{A}_1 \\ \mathbf{A}_2\end{bmatrix}$ such that $\mathbf{SA} = 0 \bmod q$ and $\mathbf{A}_2$ is invertible mod q. Equivalently, we can sample $\mathbf{A}_1$ uniformly from $\mathbb{Z}_q^{k \times m}$, and then set $\mathbf{A}_2 = -p\mathbf{S}_1\mathbf{A}_1 \bmod q$. We re-sample $\mathbf{A}_1$ if $\mathbf{A}_2$ is not invertible mod q. The SIS lattice defined by $\mathbf{A}$ is:

$$\mathcal{L} := \{(\mathbf{u}, \mathbf{v}) : \mathbf{uA}_1 + \mathbf{vA}_2 = 0 \bmod q\},$$

where $\mathbf{S}$ is a short trapdoor basis for this lattice. Note that the procedure above is a standard construction for the SIS problem, except that we have a factor of p on $\mathbf{S}_1$. We will show the equivalence between our construction and the standard SIS problem in the next subsection.

It is perhaps more convenient to look at a $k \times m$ matrix $\mathbf{B} \coloneqq \mathbf{A}_1(-\mathbf{A}_2)^{-1} \bmod q$. With $\mathbf{B}$, the lattice $\mathcal{L}$ can be interpreted as

$$\mathcal{L} \coloneqq \{(\mathbf{u}, \mathbf{v}) : \mathbf{u}\mathbf{B} = \mathbf{v} \bmod q\},$$

with a Learning with Error (LWE) basis

$$\mathbf{P} = \begin{bmatrix} 0 & q\mathbf{I}_m \\ \mathbf{I}_k & \mathbf{B} \end{bmatrix}$$

that allows for efficient sampling.

Signing: We model the hash function H as a random oracle that outputs uniformly over $\mathbb{Z}_p^n$. This allows us to generate random elements $\mathbf{m}_p \in \mathbb{Z}_p^n$ from a message digest μ. We write $\mathbf{m}_p \coloneqq (\mathbf{u}_p, \mathbf{v}_p)$, with $\mathbf{u}_p \in \mathbb{Z}_p^k$ and $\mathbf{v}_p \in \mathbb{Z}_p^m$.

The next step is to sample a vector $(\mathbf{u}_1, \mathbf{v}_1)$ from $\mathbf{P}$ such $\mathbf{u}_1 \equiv \mathbf{u}_p \bmod p$. To do so, one can simply sample a vector $\mathbf{r}$ from a discrete Gaussian distribution χ_σ^k. Then, compute $\mathbf{u}_0 = p\mathbf{r}$, $\mathbf{u}_1 = \mathbf{u}_0 + \mathbf{u}_p$, and then find a lattice vector whose "s-side" is $\mathbf{u}_1$ by setting $\mathbf{v}_1 = \mathbf{u}_1\mathbf{B} \bmod q$. As such, $(\mathbf{u}_1, \mathbf{v}_1)$ is a vector in the lattice, with $\mathbf{u}_1 \equiv \mathbf{u}_p \bmod p$.

An alternative way to view the above procedure is to generate a random vector $(\mathbf{r}, \mathbf{r}\mathbf{B} \bmod q)$ in the lattice. By definition, the matrix $[\mathbf{I}_k|\mathbf{B}]$ is a basis of a sub-lattice of $\mathcal{L}(\mathbf{P})$. Also, since $\mathbf{r}$ is sampled from a discrete Gaussian distribution, this random vector can be viewed as an output of a GPV sampler [14] over $\mathcal{L}([\mathbf{I}_k|\mathbf{B}])$. If σ is greater than the smoothing parameter of $\mathcal{L}([\mathbf{I}_k|\mathbf{B}])$, the vector $\mathbf{r}([\mathbf{I}_k|\mathbf{B}])$ will be uniform over $\mathcal{L}([\mathbf{I}_k|\mathbf{B}])$ and a discrete Gaussian over $\mathbb{Z}^n$. Then we take this vector modulo q to obtain the exact output vector.

Since $\mathbf{v}_1$ is discrete Gaussian over $\mathbb{Z}^n$, it will have random coefficients modulo p, and therefore will not meet the congruence condition. To complete the process, we need to micro-adjust $\mathbf{v}_1$ so that the t-side also meets the congruence condition; in the meantime we do not want to break the congruence condition on the s-side. We use the secret basis $\mathbf{S} = [p\mathbf{S}_1|\mathbf{I}_m]$ to achieve this goal. Let $\mathbf{a} = \mathbf{v}_p - \mathbf{v}_1 \bmod p$. We compute $(\mathbf{u}_2, \mathbf{v}_2) = \mathbf{a}\mathbf{S} = (p\mathbf{a}\mathbf{S}_1, \mathbf{a})$. Note that $(\mathbf{u}_2, \mathbf{v}_2) \equiv (0, \mathbf{a}) \bmod p$ by construction, and $(\mathbf{u}_2, \mathbf{v}_2)$ is a vector in the lattice.

The final signature is $(\mathbf{u}, \mathbf{v}) = (\mathbf{u}_1, \mathbf{v}_1) + (\mathbf{u}_2, \mathbf{v}_2)$. It is easy to see that $(\mathbf{u}, \mathbf{v})$ remains in the lattice as long as $\|\mathbf{v}\|_\infty < q/2$. On the other hand, we have

$$\mathbf{u} = \mathbf{u}_1 + \mathbf{u}_2 = \mathbf{u}_1 \equiv \mathbf{u}_p \bmod p$$

and

$$\mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2 \equiv \mathbf{v}_1 + \mathbf{v}_p - \mathbf{v}_1 \equiv \mathbf{v}_p \bmod p.$$

Therefore, $(\mathbf{u}, \mathbf{v})$ is a valid signature for our scheme.

3.2 Rejection sampling

As stated before, a candidate signature $(\mathbf{u}, \mathbf{v})$ leaks information about the secret key $\mathbf{S}$. To seal this leak one need to use the rejection sampling technique. The efficiency of the above scheme relies heavily on how often one will need to reject a signature. As a proof of concept, we will show how rejection sampling can be used to seal information leakage here. We will give a more efficient instantiation in Section 4, which uses bimodal Gaussian distribution.

Rejection sampling on $\mathbf{u}$. Recall that $\mathbf{u} = p(\mathbf{r} + \mathbf{aS}_1) + \mathbf{u}_p$. Since both p and $\mathbf{u}_p$ are publicly known, we need to seal the leakage of $\mathbf{S}_1$ from $\mathbf{b} := \mathbf{r} + \mathbf{aS}_1$. Also recall that χ_σ^k is the distribution for $\mathbf{r}$. This situation is exactly analogous to the one handled by rejection sampling in [23].

Rejection sampling on $\mathbf{v}$. On the t-side, we do not require rejection sampling. We have $\mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2$. First, $\mathbf{v}_1 = (p\mathbf{r} + \mathbf{u}_p)\mathbf{B}$, which is not linked to the secret key $\mathbf{S}_1$. Second, $\mathbf{v}_2 = (\mathbf{v}_1 - \mathbf{v}_p) \bmod p$ is also not linked to any secret key.

Another way of saying this is that rejection sampling is not required for the t-side due to the fact that the "secret key" corresponding to the t-side is actually $\mathbf{I}_m$. In fact, we can write $\mathbf{v} = \mathbf{v}_1 + \mathbf{aS}_2$ where $\mathbf{S}_2$ happens to be $\mathbf{I}_m$. As we shall see in the next section, we still need to use rejection sampling to seal the leakage for $\mathbf{S}_2$ when an alternative secret matrix replaces $\mathbf{I}_m$.

Nonetheless we do need to restart if $\|\mathbf{v}\|_\infty$ becomes too large and causes a wrap-around mod q. When this occurs, the congruent condition is broken after mod q reduction.

Alternatives. In our construction we choose to do rejection sampling so that $\|\mathbf{v}\|_\infty$ does not cause any wrap-around. We chose this approach despite the following two alternatives. First, the signer can send a helper indicating to the verifier the coefficients where wraparound occurred. This can be seen as a reconciliation approach used in (R)LWE-based key exchange methods [31, 28, 4]. We do not adopt this solution as it would increase the signature size.

Second, since the wrap-around only occurs with a low probability, we can let the verifier accept the signature based on a fuzzy matching: accept the signature when the majority of the coefficients on the t-side meet the congruent condition. This promising method may weaken our security since it makes forgery easier. For conservative purpose we do not consider this approach.

3.3 Signature compression

There are three sources of compression. First, one can effectively store only the "s-side" of the vector instead of the whole vector, so long as the vector is in $\mathcal{L}$. In other words, given $\mathbf{u}$, the verifier is able to reconstruct $\mathbf{v} = \mathbf{uB} \bmod q$.

Second, the verifier is able to reconstruct $\mathbf{u} = p\mathbf{b} + \mathbf{u}_p$ from $\mathbf{b}$ as both p and $\mathbf{u}_p$ are publicly known. So only $\mathbf{b}$ is required for verification.

Finally, since $\mathbf{b}$ follows a discrete Gaussian distribution after the rejection sampling, one can use code based compression techniques to reduce the space requirement for $\mathbf{b}$.

The final signature is a k-dimensional discrete Gaussian vector that allows for Hoffman coding. The size of the final signature is $k(\log(\sigma) + 2)$.

Algorithm 1 Signing Algorithm

Input: Message μ; Public key $\mathbf{B}$; Secret key $\mathbf{S}_1$; Distribution χ_σ
Input: Parameters k, m, p, q, M
Output: A signature $\mathbf{b}$ for message μ
1: $(\mathbf{u}_p, \mathbf{v}_p) = \text{Hash}(\mu|\mathbf{B})$
2: $\mathbf{r} \leftarrow \chi_\sigma^k$;
3: $\mathbf{u}_1 = p\mathbf{r} + \mathbf{u}_p$; $\mathbf{v}_1 = \mathbf{u}_1\mathbf{B} \bmod q$
4: $\mathbf{a} = \mathbf{v}_p - \mathbf{v}_1 \bmod p$
5: $\mathbf{v} = \mathbf{v}_1 + \mathbf{a}$;
6: **if** $\|\mathbf{v}\|_\infty \geq q/2$ **then** go to step 2 **end if**
7: **return** $\mathbf{b} = (\mathbf{r} + \mathbf{aS}_1)$ with probability $1/\left(M \exp\left(\frac{-2\langle \mathbf{b}, \mathbf{aS}_1\rangle + \|\mathbf{aS}_1\|^2}{2\sigma^2}\right)\right)$
8: go to step 2

Algorithm 2 Verification Algorithm

Input: Message μ; Public key $\mathbf{B}$; Signature $\mathbf{b}$;Parameters p, q
Output: Accept or Reject the signature
1: $(\mathbf{u}_p, \mathbf{v}_p) = \text{Hash}(\mu|\mathbf{B})$
2: $\mathbf{u} = p\mathbf{b} + \mathbf{u}_p$
3: **if** $\|\mathbf{u}\|_\infty \geq q/2$ **then** Reject **end if**
4: $\mathbf{v} = \mathbf{uB} \bmod q$
5: **if** $\mathbf{v} \not\equiv \mathbf{v}_p \bmod p$ **then** Reject **end if**
6: **return** Accept

3.4 Security

For the security of the public key, it is easy to see that the ability to find the secret key (or merely a short enough vector that allows for forging) from a public key can be reduced to the ability to solve an SIS problem. In this section we are mainly focused on the difficulty of forging signatures.

To quantify the difficulty of forgery, let us first introduce the learning with truncation problem.

Definition 3 ($\mathbf{LWT}_{q,p,n,m}$). *Let q, p, n, m be positive integers with p co-prime to q. Sample uniformly at random a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a vector $\mathbf{s} \in \mathbb{Z}_q^n$; compute $\mathbf{b} = \mathbf{sA} \bmod q \bmod p$; the decisional LWT problem is: given two pairs*

$(\mathbf{A}, \mathbf{b})$ *and* $(\mathbf{A}, [\mathbf{u}]_p)$ *where* $\mathbf{u}$ *is sampled uniformly at random in* $\mathbb{Z}_q^n$*, distinguish those two pairs. The computational problem is: given* $(\mathbf{A}, \mathbf{b})$*, find* $\mathbf{s}$*.*

As mentioned earlier, this LWT problem can be viewed as an inverse of the LWR problem. Here we show the reduction between the problems.

Lemma 1. *Choose a pair* (p, q) *such that both* p *and* $r \equiv p^{-1} \mod q$ *are on the order of* $\sqrt{q}$*. Then, if there exists an algorithm* $\mathcal{A}$ *that solves the computational LWT with parameters* q, p, n, m *for any input* $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_p^m$*, there exists another algorithm* $\mathcal{B}$ *that solves the computational LWR with parameters* q, r, n, m*, with* $(\mathbf{A}', \mathbf{b}')$ *for* $\mathbf{A}'$ *sampled uniformly at random from* $\mathbb{Z}_q^{n \times m}$*.*

We sketch the proof here.

Proof. Suppose algorithm $\mathcal{A}$ is able to solve the LWT problem, that is, given $(\mathbf{A}, \mathbf{b})$ it finds a lattice vector $\mathbf{v}$ such that

- $\mathbf{v} = \mathbf{b} \mod p$, and
- $\mathbf{v} = \mathbf{tA} \mod q$ for some $\mathbf{t}$.

Then, we can build an oracle that, upon input $(\mathbf{A}, \mathbf{b})$, it finds vectors $\mathbf{u}$ and $\mathbf{t}$, such that

$$\mathbf{v} + p\mathbf{u} = \mathbf{tA} \mod q,$$

for some $\|\mathbf{u}\|_\infty \leq \lfloor \frac{q}{2p} \rfloor < r$.

Given an input of an LWR instance $(\mathbf{A}', \mathbf{b}')$, algorithm $\mathcal{B}$ sets $\mathbf{A} = p\mathbf{A}'$, $\mathbf{b} = \mathbf{b}'$ and $r = p^{-1} \mod q$; then $\mathcal{B}$ invokes $\mathcal{A}$ with input $(\mathbf{A}, \mathbf{b})$. Since $\mathbf{A}'$ is drawn from uniform, and p is co-prime with q, $\mathbf{A}$ is also uniform over $\mathbb{Z}_q^{n \times m}$. Also, since $\|\mathbf{b}\|_\infty \leq p$ by design, $(\mathbf{A}, \mathbf{b})$ will be a legitimate input to $\mathcal{A}$. Therefore, $\mathcal{A}$ will find $\mathbf{u}$ and $\mathbf{t}$ such that $\mathbf{b} + p\mathbf{u} = \mathbf{tA} \mod q$, which is

$$r\mathbf{b}' + \mathbf{u} = \mathbf{tA}' \mod q \qquad \text{and} \qquad \mathbf{b}' = \lfloor \mathbf{tA}' \mod q \rfloor_r$$

Therefore, $\mathbf{t}$ is the solution to the computational LWR problem. ■

Now we are ready to quantify the hardness of the forgery.

Theorem 1 (Unforgeability). *Let* $\mathbf{B}$ *be a public key generated as per our scheme with parameters* q, p, n, m*. For any* new *input message* μ*, if an adversary* $\mathcal{A}$ *is able to forge a signature with a non-negligible probability* ρ*, then there is an algorithm* $\mathcal{B}$ *that solves* $LWT_{q,p,k,m}$ *with the same probability.*

Proof. First, we have modeled the hash function H as a random oracle that outputs uniformly over $\mathbb{Z}_p^n$. In addition, the forger is asked to sign on a message that he has not seen before. Hence, if algorithm $\mathcal{B}$ is able to forge a signature for every legitimate input μ with non-negligible probability, it must be that $\mathcal{A}$ is able to forge a signature for any legitimate $\mathbf{m}_p = H(\mu|\mathbf{B})$. In the meantime, any *new* "mod p" vector will look like a legitimate hash output from the forger's point of view.

Next, we claim that $\mathbf{B}$ is indistinguishable from a matrix randomly and uniformly chosen from $\mathbb{Z}_q^{k\times m}$. This follows from the fact that $\mathbf{A}$ is indistinguishable from a matrix randomly and uniformly chosen from $\mathbb{Z}_q^{n\times m}$. Recall $\mathbf{B} = \mathbf{A}_1(-\mathbf{A}_2)^{-1} \bmod q$ and $\mathbf{A} = \begin{bmatrix}\mathbf{A}_1\\ \mathbf{A}_2\end{bmatrix}$.

Therefore, given an LWT instance $(\mathbf{A}', \mathbf{b}')$, the forger cannot distinguish $\mathbf{A}'$ from a legitimately generated public key; it also cannot distinguish $\mathbf{b}'$ from a legitimately generated public message digest. As a result, it will return a signature vector $\mathbf{v}$ which will pass the verification test with probability ρ. From $\mathbf{v}$ it is easy to extract the solution to the LWT problem. ■

We remark that to have such a tight reduction from the forgery to LWR/LWT we will have required a rather large p, on the order of $\sqrt{q}$, which makes this scheme less efficient. As we will see in next section, our efficient instantiation uses practical parameters that are derived from best-known attacks (this is also the design principle for most practical lattice-based signatures, except for [3]). For this purpose we will choose a small p that allows for efficient rejection sampling.

Strong unforgeability. One subtlety in the (standard) unforgeablity notion is that the forger is asked to sign messages that have never been previously signed. The notion of strong unforgeability, however, requires an attacker to be unable to forge a signature on a message, even if a set of signatures of this same message are given. This is not captured by the above theorem. Indeed, here we show a modification that allows strong unforgeability to be achieved.

As shown in [17], for a given message digest $\mathbf{m}_p$, all candidate signatures associated with this message digest are *short* vectors within the intersection of the original lattice and $p\mathbb{Z}^n + \mathbf{m}_p$. Therefore, the task of forgery becomes finding a short vector in the original lattice that meets the length requirement and the congruence mod p requirement. This is roughly the reduction to the approx-CVP in [17].

Now, suppose that the attacker is given a list of signatures on a same message digest, then, it becomes easier (compared to without this list) for the attacker to find another short vector in this lattice, that is, generating a new signature on this same message. However, we note that any linear combination of such signatures is highly unlikely to also satisfy the correct mod p congruence conditions.

In general, our solution to achieving strong unforgeability is to include a random salt in the hash when generating the message digest; this salt will be part of the signature and used during verification. This ensures that it is highly improbable, (probability $(1/p)^{2n}$ for each message), that the same message digest will occur more than once. Note that this is also the same technique that provides similar functionalities for GPV based signatures [14].

Nevertheless, as the strong unforgeability model is sometimes too strong for practical use (i.e., the attacker doesn't need to forge a new signature since it has already got a list of them on a same message), we leave out this salt in our efficient instantiation to minimize signature size.

4 A practical instantiation with an NTRU lattice

In the previous section we presented an inefficient modular lattice signature scheme based on the SIS/LWT which requires $n \approx m\log(m)$. Even if we use the ring-SIS version the scheme is still somewhat inefficient as it requires $n \approx \log(m)$ - the reduction of a factor of m comes directly from the use of the ring. A natural way to improve its efficiency is to relax the requirement of $n \approx \log(m)$ (This will make the underlying (ring) SIS problem easier, so we will derive parameters from the best known attacks).

For example we can reduce n to $2m$ (2 in the case of ring-SIS). This makes $\mathbf{A}_1$ a square matrix which causes another issue:

$$p\mathbf{S}_1\mathbf{A}_1 + \mathbf{I}_m\mathbf{A}_2 = 0 \bmod q.$$

When $\mathbf{A}_1$ is a square matrix and invertible, one can easily recover $\mathbf{S}_1$ from $\mathbf{A}_1$ and $\mathbf{A}_2$.

A naive remedy is to set $\mathbf{A}_2$ to be private too, and therefore we will have a secret matrix $[p\mathbf{S}_1|\mathbf{A}_2]$ and a public matrix $\begin{bmatrix}\mathbf{A}_1\\ \mathbf{I}_m\end{bmatrix}$ which also satisfies the above equation without giving away $\mathbf{S}_1$. This seemingly plausible solution poses another challenge: we are not able to perform a micro-adjustment on the "t-side" of the vector anymore, as now $\mathbf{A}_2$ is no longer small. If we perform the same micro-adjustment as before, the coefficients of $\mathbf{u}_2$ will explode and will always cause wrap-around over q.

Hence, the final solution to the above problem is to have a small and private $\mathbf{A}_2$. The final key generation becomes finding such $\mathbf{A}_2$ and an invertible $\mathbf{S}_1$, and setting $\mathbf{A}_1 = \mathbf{A}_2(p\mathbf{S}_1)^{-1} \bmod q$. This, not surprisingly, yields an NTRU lattice. In the following, we will slightly change the notation: $\mathbf{H} \coloneqq \mathbf{A}_1$, $\mathbf{G} \coloneqq \mathbf{A}_2$ and $\mathbf{F} \coloneqq \mathbf{S}_1$.

4.1 Overview

In the following we will work over the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^N+1)$. Our scheme also works over other rings, such as $\mathbb{Z}_q[x]/(x^N-1)$ with minor modification. Let $f(x)$, $g(x)$ and $h(x)$ be 3 polynomials in $\mathcal{R}_q$, where $f(x)$ and $g(x)$ have very small coefficients; $h(x) = p^{-1}g(x)f^{-1}(x)$. We express by $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{h}$ the vector form of the polynomials. Also let $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{H}$ be the matrix obtained from nega-cyclic rotations. The NTRU lattice with regard to h is defined as

$$\mathcal{L}_h = \{(u,v) \in \mathcal{R}_q^2 : uh = v\}$$

or rather, the vector/matrix form:

$$\mathcal{L}_h = \{(\mathbf{u},\mathbf{v}) : \mathbf{u}\mathbf{H} = \mathbf{v} \bmod q\}$$

where there exists a public basis $\mathbf{P} = \begin{bmatrix}0 & q\mathbf{I}_N\\ \mathbf{I}_N & \mathbf{H}\end{bmatrix}$ and a secret generator $[p\mathbf{F}|\mathbf{G}]$. We also require $g(x)$ to be invertible over $\mathcal{R}_p$, which is the same as $\mathbf{G}$ being invertible mod p.

The rest of the scheme is almost identical to the one presented in the previous section, except for two differences.

First, we use a bimodal Gaussian distribution to improve the acceptance rate. To cope with this modification, we set $p = 2$ so that the change of signs in $\mathbf{b} = \mathbf{r} \pm \mathbf{af}$ will vanish after reduction modulo p.

Second, we use $[p\mathbf{F}|\mathbf{G}]$ rather than $[p\mathbf{S}_1|\mathbf{I}_m]$ to perform the micro-adjustment. This modification does raise another issue: the "t-side" vector during the signing procedure will contain information about $\mathbf{G}$. To be precise, the "t-side" vector will be $\mathbf{v} := \mathbf{v}_1 \pm \mathbf{ag}$ where $\mathbf{v}_1$ is indistinguishable from uniform over $\mathcal{R}_q$, $\mathbf{a}$ is uniform over $\mathbb{Z}_p^N$. We will need to perform rejection sampling to seal the leakage of information about $\mathbf{g}$. As shown in [17], after rejection sampling, the distribution of $\mathbf{v}$ will be computationally indistinguishable from uniform over $(-\frac{q}{2} + B, \frac{q}{2} - B)$ for a parameter B which depends on $\mathbf{a}$, $\mathbf{g}$ and q.

To avoid confusion, we will use M_s to denote the rejection rate for the s-side, M_t for the t-side, and M for the overall rate.

4.2 The scheme

Key generation : The key generation algorithm is shown in Algorithm 3. We use

Algorithm 3 Key Generation Algorithm

Input: Parameters N, p, q, d
Output: Public key $\mathbf{h}$ and secret key $(p\mathbf{f}, \mathbf{g})$
1: $\mathbf{f} \leftarrow T(d+1, d)$
2: **if** $\mathbf{f}$ is not invertible mod q **then** go to step 1 **end if**
3: $\mathbf{g} \leftarrow T(d+1, d)$
4: **if** $\mathbf{g}$ is not invertible mod p **then** go to step 3 **end if**
5: $\mathbf{h} = \mathbf{g}/(p\mathbf{f}) \bmod q$
6: **return** $\mathbf{h}$, $\mathbf{g}$ and $\mathbf{f}$

the classical NTRU flat form (non-product form, cf. [18]) keys with a pre-fixed number of $+1$s and -1s. Here, $T(d_1, d_2)$ is a set of trinary polynomials of degree less than N, where there are exactly d_1 positive coefficients and d_2 negative coefficients. One can choose thicker keys for a higher level of security. Since we require both $\mathbf{f}$ and $\mathbf{g}$ to be invertible, we have set $f(1) = g(1) = 1$.

Remark 3. In BLISS [9], there is an extra rejection sampling process on keys $\mathbf{f}$ and $\mathbf{g}$ during key generation, so that $\|\mathbf{af}\|$ is reasonably bounded for efficient rejection sampling on signatures. We do not adopt this process. Rather we perform rejection sampling on $\|\mathbf{af}\|$ during the signing procedure.

Signing algorithm : We highlight the differences between this signing algorithm and the one described in previous section.

Algorithm 4 Signing Algorithm

Input: Message μ; Public key $\mathbf{h}$; Secret key $\mathbf{f}$ and $\mathbf{g}$; Distribution χ_σ
Input: Parameters N, p, q, M_s, B_s, B_t
Output: A signature $\mathbf{b}$ for message μ

1: $(\mathbf{u}_p, \mathbf{v}_p) = \text{Hash}(\mu|\mathbf{h})$
2: $\mathbf{r} \leftarrow \chi_\sigma^N$, $b \leftarrow \{0, 1\}$
3: $\mathbf{u}_1 = p\mathbf{r} + \mathbf{u}_p$; $\mathbf{v}_1 = \mathbf{u}_1\mathbf{h} \bmod q$
4: $\mathbf{a} = (\mathbf{v}_p - \mathbf{v}_1)/\mathbf{g} \bmod p$
5: **if** $\|\mathbf{af}\|_2 > B_s$ or $\|\mathbf{ag}\|_\infty > B_t$ **then** go to step 2 **end if**
6: $\mathbf{v} = \mathbf{v}_1 + (-1)^b\mathbf{ag}$;
7: **if** $\|\mathbf{v}\|_\infty > q/2 - B_t$ **then** go to step 2 **end if**
8: **return** $\mathbf{b} = (\mathbf{r} + (-1)^b\mathbf{af})$ with probability $1/\left(M_s \exp\left(-\frac{\|\mathbf{af}\|}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{b}, \mathbf{af}\rangle}{\sigma^2}\right)\right)$
9: go to step 2

First, there is a factor of $\mathbf{g}^{-1} \bmod p$ for step 4, which is there to ensure the congruence condition for the t-side.

Second, in step 5, we check the norm requirements for $\mathbf{af}$ and $\mathbf{ag}$. This is to ensure that the rejection samplings in the followed steps deliver the desired acceptance rate.

Third, in step 7, rejection sampling is performed on the t-side, parameterized by an additional integer B_t. The distribution of the t-side vector will be uniform within the interval $(-\frac{q}{2} + B_t, \frac{q}{2} - B_t)$.

Finally, unlike the scheme in previous section, here we have

$$(\mathbf{u}, \mathbf{v}) = (\mathbf{u}_1, \mathbf{v}_1) + (-1)^b(\mathbf{u}_2, \mathbf{v}_2)$$

for a random bit b. This makes the raw distribution of $\mathbf{b} := (\mathbf{r} + (-1)^b\mathbf{af})$ a bimodal Gaussian distribution. As stated before, one can achieve a much higher acceptance rate for this distribution. Note that in the initial construction of BLISS [9], the bimodal Gaussian distribution makes a signature sometimes unverifiable due to the odd modulus q. BLISS solved this problem by moving the modulus from q to $2q$. We solve this problem by setting $p = 2$. It follows that $\mathbf{v} \equiv \mathbf{v}_1 + (-1)^b(\mathbf{v}_p - \mathbf{v}_1) \equiv \mathbf{v}_p \bmod 2$.

Algorithm 5 Verification Algorithm

Input: Message μ; Public key $\mathbf{h}$; Signature $\mathbf{b}$;Parameters p, q, B, σ, N
Output: Accept or Reject the signature

1: $(\mathbf{u}_p, \mathbf{v}_p) = \text{Hash}(\mu|\mathbf{h})$
2: $\mathbf{u} = p\mathbf{b} + \mathbf{u}_p$
3: **if** $\|\mathbf{u}\|^2 > p^2\sigma^2 N$ **then** Reject **end if**
4: $\mathbf{v} = \mathbf{uh} \bmod q$
5: **if** $\mathbf{v} \not\equiv \mathbf{v}_p \bmod p$ or $\|\mathbf{v}\|_\infty > q/2 - B$ **then** Reject **end if**
6: **return** Accept

4.3 Security

A similar reduction to the approximate shortest vector problem can be applied here, except that we need to adjust the approximation parameter for the γ-SVP because $\lambda_1 = \|(\mathbf{f}, \mathbf{g})\|$ in the NTRU lattice is smaller than the Gaussian heuristic length. We omit the details.

5 Batch verification

The modular lattice signature scheme presented here allows for batch verification. This is because, as stated in the introduction, the sum of signatures, after lifting to the integers, is still a valid lattice vector that satisfies the mod p congruence condition.

However, in order to fully utilize this functionality, it appears at first that one will need to send the whole lattice vector as the signature. In other words, one cannot merely send the "s-side" of the vector. To see why this is the case, suppose that for two signatures $(\mathbf{u}, \mathbf{v})$ and $(\mathbf{u}', \mathbf{v}')$ corresponding to messages $(\mathbf{u}_p, \mathbf{v}_p)$ and $(\mathbf{u}'_p, \mathbf{v}'_p)$, one computes

$$(\mathbf{v} + \mathbf{v}') \bmod q = (\mathbf{u} + \mathbf{u}')\mathbf{h} \bmod q$$

The difficulty is that $(\mathbf{v} + \mathbf{v}')$ will, with high probability, cause a wraparound mod q, as $\|\mathbf{v} + \mathbf{v}'\|_\infty \lessapprox q - 2B_t$. Thus one will recover $(\mathbf{v} + \mathbf{v}') \bmod q$ rather than $\mathbf{v} + \mathbf{v}'$. When this occurs,

$$(\mathbf{v} + \mathbf{v}') \bmod q \bmod p \neq (\mathbf{v}_p + \mathbf{v}'_p) \bmod p$$

and the verification will fail.

One way to solve this issue is to send both the "s-side" and the "t-side" of the vector. Then one recovers $\mathbf{u} + \mathbf{u}'$ and $\mathbf{v} + \mathbf{v}'$ over the integers. The mod p relationship can be checked from this, and then the lattice relation mod q can be checked. As a trade-off, one will have to send 2 elements in $\mathcal{R}_q$ for each signature. This increases the size of a signature.

We can actually do efficient batch verification with a much smaller cost. We can send merely the "t-side" of the vectors. Then the sum of the t-side vectors can be computed over the integers, and the congruence mod p can be checked. Then, multiplying by h^{-1} and reducing mod q will reveal the sum of the "s-side" of the vectors mod q. Signature aggregation works so long as the sum of the "s-side" vectors mod q identically equals the sum over the integers, that is, does not result in any wrap-around modulo q. Since the "s-side" vectors are Gaussian distributed with a variance σ much smaller than q, we are able to sum quite a few s-side vectors without a wrap-around mod q.

To be precise, suppose we want to verify k signatures in one batch. Since a sum of k samples from χ_σ is also a Gaussian with variance $\sqrt{k}\sigma$, we know that the maximum absolute value of the coefficients, i.e, $\|\sum_k \mathbf{u}_i\|_\infty$, will be bounded above by $\sqrt{k}\tau\sigma$ (recall that τ is the Gaussian tail-cutting parameter). Therefore,

having $\sqrt{k}\tau\sigma \leq q/2$ will eliminate wrap-around. That is, we are able to batch verify

$$k = \lfloor \Big(\frac{q}{2\tau\sigma} \Big)^2 \rfloor$$

signatures in one batch. For our parameter choices, to be shown in the next section, we have $k = 529$, $\sigma = 107$ and $\tau = 13.3$. See Algorithm 6, below, for the batch verification algorithm.

Algorithm 6 Batch Verification Algorithm

Input: Messages $\{\mu_i\}$; Public key $\mathbf{B}$; Signature $\{\mathbf{v}_i\}$;Parameters p, q, B, k, σ
Output: Accept or Reject the signature

1: $(\mathbf{u}_{p,i}, \mathbf{v}_{p,i}) = \text{Hash}(\mu_i|\mathbf{B})$
2: **if** $\|\mathbf{v}_i\|_\infty > q/2 - B_t$ **then** Reject **end if**
3: $(\mathbf{u}_p, \mathbf{v}_p) = 0$; $\mathbf{v} = 0$
4: **for** $i \in [k]$ **do**
5: $\quad (\mathbf{u}_p, \mathbf{v}_p)$ += $(\mathbf{u}_{p,i}, \mathbf{v}_{p,i})$
6: $\quad \mathbf{v}$ += $\mathbf{v}_i$
7: **end for**
8: $\mathbf{u} = \mathbf{v}\mathbf{h}^{-1} \bmod q$
9: **if** $\|\mathbf{u}\|_\infty > \sqrt{k}\tau p\sigma$ **then** Reject **end if**
10: **if** $(\mathbf{u}, \mathbf{v}) \not\equiv (\mathbf{u}_p, \mathbf{v}_p) \bmod p$ **then** Reject **end if**
11: **return** Accept

5.1 Attack and proof for batch verification

Here is an potential attack on batch setting, which performs better than forging a single signature directly. For a set of message digests $\{\mathbf{u}_p^{(i)}, \mathbf{v}_p^{(i)}\}$ for $1 \leq i \leq k$, do the following:

- for each $\mathbf{v}_p^{(i)}$, find a random vector $\mathbf{v}_1^{(i)}$ such that $\mathbf{v}_1^{(i)} = \mathbf{v}_p^{(i)} \bmod p$;
- set $\mathbf{V} = \sum_i^k \mathbf{v}_1^{(i)}$; $\mathbf{V}$ meets the congruent condition by design;
- compute $\mathbf{U} = \mathbf{V}\mathbf{h}^{-1}$,
- Since we allows $(\mathbf{U}, \mathbf{V})$ to be reasonably large, we can simply use the public key/basis $(\mathbf{I}, p\mathbf{H}^{-1})$ for the micro-adjustments. Suppose the micro adjustment vector is $(\mathbf{U}_0, \mathbf{V}_0)$
- Write $\mathbf{V}_0$ as a sum of k vectors $\{\mathbf{v}_0^{(i)}\}$
- Publish $\mathbf{v}_i = \mathbf{v}_0^{(i)} + \mathbf{v}_1^{(i)}$ as the signatures.

In short, The attacker finds a large vector $(\mathbf{U}, \mathbf{V})$ in the lattice, congruent to the sum of messages mod p. In addition, $\mathbf{V}$ can be written as a sum of k different $\mathbf{v}^{(i)}$'s such that $\mathbf{v}^{(i)}$ is congruent to $\mathbf{v}_p^{(i)}$ for each message mod p. In the meantime, the $\mathbf{U}$ vector also meet the congruence condition; while the attacker doesn't need to find individual $\mathbf{u}^{(i)}$'s. In the meantime, for sufficiently large k,

the size of $(\mathbf{U}, \mathbf{V})$ will be acceptable. Hence, the attack can claim that each such small vector is a signature for a given message, as collectively they can be batch verified, although each individual verification will fail.

Note that for this attack to work, k needs to be large. For properly chosen k this attack will fail. The intuition is that, when k is small enough, the sum of k valid signatures will remain a short vector in the lattice so that the root Hermite factor with respect to this lattice/vector is still small (although it will be larger than in a single verification setting). In other words, if the attacker is able to find a vector $(\mathbf{U}, \mathbf{V})$ sufficiently small, he is also able to find an approximate shortest vector in the lattice (with a root Hermite factor slightly larger than the single verification case, see Section 6.2 for an analysis of the lattice strength of both single and batch verifications against forgery).

6 Parameters and implementations

6.1 Parameter derivation

The script that we used to generate the parameters is available at [2]. Here we give an example of how to derive parameters for 128 bits security.

We use $N = 512$ which allows for efficient FFT (together with an FFT-friendly modulus $q = 2^{16} + 1$). We also set $p = 2$, which is relatively prime to q in both settings, and also enables the use of the bimodal Gaussian. The secret polynomials $\mathbf{f}$ and $\mathbf{g}$ are sparse trinary polynomials with roughly $2d_f + 1 = 2d_g + 1 = 0.3N \approx 155$ number of non-zero coefficients. This creates an NTRU lattice with unique shortest vectors. This is also the NTRU lattice used in [9]. In next subsection we show the lattice strength against recovering those unique shortest vectors.

The next most important parameter in our scheme is σ. We need to have σ small enough to avoid wrap-around mod q on the s-side when performing batch verification. This requires the following upper bound for $\sigma < \frac{q}{2\tau p}$. As remarked before, a smaller σ produces a more compact signature, in the single signature verification scenario, but at the cost of worsening the rejection rate. Thus we may chose smaller σ to obtain a slower signature algorithm with smaller (single) signature size. Alternatively, we can trade the size for faster signing. Here, we follow the BLISS scheme by setting $\sigma = 107$. This value seems to give a good signature size while maintaining an acceptable rejection rate.

Rejection rate. Next, we calculate the rejection rate. As shown in Eq. 1, the rejection rate parameter M_s depends on σ and $\max_{\mathbf{f}}(\|\mathbf{af}\|) = B_s$. From Eq. 1, if $B_s \approx 2\sigma$, then we can expect a rejection parameter for the "s-side" of $M_s = e^2 \approx 7.4$. Assuming $\mathbf{a}$ is uniformly distributed in $\mathbb{Z}_p^N$, our implementation shows that $\text{Prob}(\|\mathbf{af}\|_2 \leq B_s) \approx 89\%$ for this choice of B_s. Therefore the overall probability of acceptance on the "s-side" is $0.89/e^2 \approx 12\%$.

For the "t-side", we simply run an exhaustive search for the optimal B_t as follows:

– Sample many $\mathbf{v}_1$ uniformly from $\mathcal{R}_q$;
– Sample many $\mathbf{a}$ uniformly from $\{-1, 0, 1\}^N$;
– Sample many $\mathbf{g}$ uniformly from $T(d+1, d)$;
– For each set of samples, compute $\mathbf{v} = \mathbf{v}_1 + \mathbf{ag}$
– find B_t which minimizes

$$M_t := 1/\operatorname{Prob}(\|\mathbf{ag}\|_\infty \leq B_t, \|\mathbf{v}\|_\infty \leq \frac{q}{2} - B_t).$$

Our test shows that with $B_t = 40$, this quantity is minimized at around 2.

$\mathcal{R}, N, q$	d_f, d_g	σ	B_s, B_t	PK size	Sig size
$\frac{\mathbb{Z}_q[x]}{x^N+1}, 512, 2^{16}+1$	77	107	215, 40	8704 bits	4608 bits

Table 1. Parameters.

Overall	M_s	Prob($\|\mathbf{af}\|_2 \leq B$)	M_t
6%	7.4	89%	2

Table 2. Acceptance rates.

6.2 Estimating the lattice strength

Security against public key attacks. The public key strength is determined by the hardness of the unique shortest vector problem of the NTRU lattice, which is related to the $2N$-th root of the following quantity:

$$\frac{\text{Gaussian Heuristic Length}}{\lambda_1} = \frac{\sqrt{2N/(2\pi e)}q^{N/(2N)}}{\|\mathbf{f}, \mathbf{g}\|_2} = \sqrt{\frac{Nq/(\pi e)}{2d_f + 2d_g + 2}}.$$

Security against forgery. We analysis the (batch) forgery attack with a parameter k. For single verification, we can simply apply the results in this section with $k = 1$.

In [17] it is shown in Section 5 that the forging a signature can be accomplished if an associated approximate closest vector problem in the intersection of the NTRU lattice, and $p\mathbb{Z}^{2N}$ can be solved. Therefore, the task of forgery can be solved by finding a vector that meets the congruence mod p requirements, and is sufficiently close to the intersection lattice to satisfy the length requirement.

This problem is harder than that of finding a short vector in the intersection lattice, and so to simplify our analysis we will use this to quantify the strength of the lattice problem. The intersection lattice is generated by the rows of the matrix

$$\begin{bmatrix} 0 & pq\mathbf{I}_N \\ p\mathbf{I}_N & p\mathbf{H}' \end{bmatrix},$$

for some appropriate $\mathbf{H}'$. We also assume that this lattice behaves like a random lattice.

Notice that the lattice is not "balanced" as $\|\mathbf{u}\|$ is significantly smaller than $\|\mathbf{v}\|$. In general, if the target is a vector $(\mathbf{u}, \mathbf{v})$, with $\mathbf{u}, \mathbf{v}$ each N-dimensional, and satisfying $\|\mathbf{u}\| \approx a\sqrt{N}$ and $\|\mathbf{v}\| \approx b\sqrt{N}$ then the optimal matrix for maximizing strength against lattice reduction attacks, that is, minimizing the ratio of the norm of the target of the Gaussian expected norm, is the $2N$ by $2N$ matrix

$$\begin{bmatrix} 0 & pq\mathbf{I}_N \\ \alpha p\mathbf{I}_N & p\mathbf{H} \end{bmatrix},$$

with α chosen so that $\alpha = b/a$.

The vector $(\alpha\mathbf{u}, \mathbf{v})$ will be a short vector in this matrix, and it is not surprising that the optimal α equalizes the lengths of the vectors $\alpha\mathbf{u}$, and $\mathbf{v}$. We omit the details justifying this.

We now determine the values of a, b in our case. As it is a sum of k vectors, with each coordinate choses from the Gaussian distribution, the expected norm of $\|\mathbf{u}\|$ will satisfy $\|\mathbf{u}\|^2 \approx p^2\sigma^2 kN$. Thus $a = p\sigma\sqrt{k}$. Also,

$$\mathbf{v} = \sum_{i=1}^{k} \mathbf{v}_i,$$

with the coordinates of each $\mathbf{v}_i$ approximately randomly and uniformly distributed between $-q/2 + B_t$ and $q/2 - B_t$. As uniformly distributed vectors in high dimensions are close to orthogonal, It follows that

$$\|\mathbf{v}\|^2 \approx \sum_{i=1}^{k} \|\mathbf{v}_i\|^2.$$

Each coordinate of $\mathbf{v}_i$ will be approximately randomly and uniformly distributed between $-q/2+B_t$ and $q/2-B_t$. Ignoring the B_t, the average squared coefficient will be approximately

$$\frac{1}{q}\int_{-q/2}^{q/2} x^2 dx = q^2/12.$$

Thus $\mathbf{v}$ will have norm $\|\mathbf{v}\|^2 \approx kq^2N/12$, so $b = q\sqrt{k/12}$.

As stated above, in our particular case $a = p\sigma\sqrt{k}$, $b = q\sqrt{k/12}$, so $\alpha = q/(p\sigma\sqrt{12})$, and the length of the target is

$$\text{Length target} \approx b\sqrt{2N} = q\sqrt{kN/6}.$$

For general, a, b, and $\alpha = b/a$, the determinant of the matrix is $\alpha^N p^{2N} q^N$, and thus the length of the Gaussian expected shortest vector is

$$\text{Gaussian Heuristic Length} = \alpha^{1/2} p q^{1/2} \sqrt{\frac{2N}{2\pi e}} = \sqrt{\frac{Npq^2}{\pi e \sigma\sqrt{12}}}$$

We thus have

$$\frac{\text{Target Length}}{\text{Gaussian Heuristic Length}} = \sqrt{\frac{\pi e \sigma k}{p\sqrt{3}}},$$

and the strength against forgery is determined by the $2N^{th}$ root of this ration, which equals.

$$\left(\frac{\pi e \sigma k}{p\sqrt{3}}\right)^{1/(4N)}.$$

	$k = 1$	$k = 529$	$k = 2000$
Public key strength $\left(\frac{\text{GH}}{\lambda_1}\right)^{\frac{1}{2N}}$	$112^{\frac{1}{2N}} = 1.0046$		
Forgery strength $\left(\frac{\|\mathbf{u},\mathbf{v}\|}{\text{GH}}\right)^{\frac{1}{2N}}$	$16^{\frac{1}{2N}} = 1.0027$	$78^{\frac{1}{2N}} = 1.0043$	$109^{\frac{1}{2N}} = 1.0046$

$k = 1$: single verification case
$k = 529$: theoretical bound without verification errors
$k = 2000$: practical bound from experiments without verification errors

Table 3. Lattice strength given by root Hermite factor

We estimate that our parameter set delivers 128 bits security against classical and quantum attackers, assuming the complexity of BKZ 2.0 using enumeration with extreme pruning [8, 13]. This is using the same metric as was used in [9] and [17].

6.3 Implementation and performance

We implemented our scheme with C. Our software is available at [2] under GPL license. We benchmarked our implementation on a dual core Intel i7-6600U processor @ 2.60GHz. Our operation system was Linux Ubuntu 16.04. We used gcc version 5.4.0.

	Single verification	Batch verification
Signature size	4608 bits	8192 bits
Public key size	≈8200 bits	≈8200 bits
Signing time	$15 \sim 20$ms	$15 \sim 20$ ms
Verification time	0.3ms	0.3 ms

Table 4. Performance

The benchmark results are given in Table 4. As mentioned previously, we observed that in practice one may perform successful batch verification for a number of signatures between 1000 to 2000, which is higher than the theoretical threshold $k = 529$.

We also note that we did not use FTT/NTT techniques to accelerate ring multiplications in signing/verification since we need to perform mod p over the integers regularly. We leave the investigation of this potential optimization to future work.

Remark 4. A "t-side" signature vector can always be stored with $16N = 8192$ bits as it needs to be smaller than $\frac{q}{2} - B_t$ in infinity norm. For public keys, an element in $\mathcal{R}_q$ with $q = 2^{16}+1$ can also be stored efficiently with 8192 bits, except for the case where one or more coefficients are equal to $q - 1$. This occurs with $1/q$ probability if the coefficient is uniformly random in $\mathbb{Z}_q$. When this happens, we need an extra of $\lceil \log_2 N \rceil = 9$ bits for each such coefficient to indicate the position of the coefficient. The final size of the public is therefore around 8200 bits.

7 Conclusion and comparison

In this paper we revisited the NTRU modular lattice signature scheme [17]. We presented an instantiation of the modular lattice signature, using bimodal Gaussian sampling [9] and the NTRU lattice [20].

Compared to the original pqNTRUSign scheme, we are able to remove an extra assumption in the original design. The procedure for generating a signature is similar, except for the method of sampling the random lattice vector. We improve both the signing speed and the signature size. We also enable signature aggregation with this new construction.

Compared to the BLISS scheme, our approach (hash-then-sign) is an entirely different approach. However, the final signatures in both schemes are discrete Gaussian vectors of similar parameters. As explained before, in our scheme we can efficiently store the s-side vector while the verifier can reconstruct the whole lattice vector during the verification. This advantage saves us from sending a helper as done in BLISS. In particular, our parameter set uses identical parameter sets for bimodal sampling as in BLISS-II. Our signature is around 400 bits less than BLISS which is exactly the storage requirement for the helper vector. In terms of the speed, our scheme has almost same rejection rate as BLISS-II on the "s-side". However, our overall speed is twice as slow as BLISS-II, due to the uniform rejection sampling on "t-side".

References

1. NSA Suite B Cryptography - NSA/CSS.
2. NTRU OpenSource Project. online. available from `https://github.com/NTRUOpenSourceProject/ntru-crypto`.
3. Erdem Alkim, Nina Bindel, Johannes A. Buchmann, and Özgür Dagdelen. TESLA: tightly-secure efficient signatures from standard lattices. *IACR Cryptology ePrint Archive*, 2015:755, 2015.
4. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 327–343, 2016.

5. Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 57–74, 2013.
6. Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 719–737, 2012.
7. Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. Report on post-quantum cryptography. National Institute of Standards and Technology Internal Report 8105, February 2016.
8. Yuanmi Chen and Phong Q Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT 2011*, pages 1–20. Springer, 2011.
9. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 40–56, 2013.
10. Léo Ducas, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - dilithium: Digital signatures from module lattices. *IACR Cryptology ePrint Archive*, 2017:633, 2017.
11. Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, pages 22–41, 2014.
12. Léo Ducas and Phong Q. Nguyen. Learning a zonotope and more: Cryptanalysis of ntrusign countermeasures. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 433–450, 2012.
13. Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 257–278. Springer, 2010.
14. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, page 197206, New York, NY, USA, 2008. ACM.
15. Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, pages 112–131, 1997.
16. Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSIGN: digital signatures using the NTRU lattice. In *Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA, April 13-17, 2003, Proceedings*, pages 122–140, 2003.
17. Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, and William Whyte. Transcript secure signatures based on modular lattices. In *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, pages 142–159, 2014.

18. Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for ntruencrypt. *IACR Cryptology ePrint Archive*, 2015:708, 2015.
19. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: a ring-based public key cryptosystem. In *Algorithmic number theory (Portland, OR, 1998)*, volume 1423 of *Lecture Notes in Comput. Sci.*, pages 267–288. Springer, Berlin, 1998.
20. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, pages 267–288, 1998.
21. Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *Advances in CryptologyASIACRYPT 2009*, page 598616. Springer, 2009.
22. Vadim Lyubashevsky. Lattice signatures without trapdoors. In *Advances in cryptology—EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Comput. Sci.*, pages 738–755. Springer, Heidelberg, 2012.
23. Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, 2012.
24. Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 577–594, 2009.
25. Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 21–39, 2013.
26. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.
27. Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *J. Cryptology*, 22(2):139–160, 2009.
28. Chris Peikert. Lattice cryptography for the internet. In *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, pages 197–219, 2014.
29. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93, 2005.
30. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *FOCS*, pages 124–134, 1994.
31. Jiang Zhang, Zhenfeng Zhang, Jintai Ding, and Michael Snook. Authenticated key exchange from ideal lattices. *IACR Cryptology ePrint Archive*, 2014:589, 2014.