Modular Lattice Signatures, revisited

Dipayan Das, Jeffrey Hoffstein, Jill Pipher, William Whyte, and Zhenfei Zhang

[1] National Institute of Technology, Durgapur, India, dasdipayan.crypto@gmail.com
[2] Brown University, Providence RI, USA, {jhoff, jpipher}@math.brown.edu
[3] Qualcomm Technologies Incorporated, USA, wwhyte@qti.qualcomm.com
[4] Algorand, USA, zhenfei@algorand.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. 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 show that this problem is equivalent to the short integer solution (SIS) problem over the corresponding lattice. In addition, we show that by replacing the uniform sampling in pqNTRUSign with a bimodal Gaussian sampling, we can further reduce the size of a signature. An important new contribution, enabled by this Gaussian sampling version of pqNTRUSign, is that we can now perform batch verification of messages signed by the same public key, which allows the verifier to check approximately 24 signatures in a single verification process.

1 Introduction

Organizations and research groups are looking for candidate algorithms to replace RSA and ECC based schemes [CJL+16,NSA,NIS] due to the threat of quantum computers [Sho94]. In 2017, the National Institute of Standards and Technology announced a call for proposals for candidate quantum-safe algorithms [NIS]. Among all candidates, lattice based solutions seem to be most promising. This is due to two main reasons. First, lattice problems are conjectured to be quantum resistant. Second, average case lattice problems (like short integer solution problems and learning with error problems) can be shown to be as hard as the worst-case lattice problems (like gap shortest vector problems and shortest independent vector problems).

On the subject of signature schemes, there exist two families of solutions which are well studied in the literature. The first one, initiated from the celebrated rejection sampling methods of [Lyu09], was followed by a sequence of improvements and variants [Lyu12,DDLL13,AAB+,DKL+18]. They are the lattice based analogues of Schnorr signatures that use Fiat-Shamir transformation. The other family of novel constructions use the GPV framework [GPV08] that follows a hash-then-sign paradigm. Examples of this construction are [DLP14,FHK+].

A new family of lattice based signature scheme is pqNTRUSign, which was originally presented in [HPS+14] as an instantiation of modular lattice signature.

The submitted version [HWZ] to NIST PQC process has a few significant modifications/improvements over [HPS+14]. In this paper, we give a formal analysis of those improvements.

Note that by the time of this revision, NIST has announced their second round of candidate algorithms, and pqNTRUSign is not included in the second round, despite the fact that no flaws were identified during the evaluation. Nonetheless we believe that pqNTRUSign is still an interesting research topic, with potential use cases that other solutions do not provide.

1.1 Modular lattice signatures

Let us briefly recall the framework of modular lattice signatures. 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 [HPS+14] 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 principle, the above framework works for generic lattices. In practice, it is natural to instantiate the scheme with NTRU lattices [HPS98] to get optimized signatures as was done in BLISS [DDLL13] and DLP [DLP14]. The structure of the NTRU lattice provides much flexibility in choosing the parameters, resulting in an optimized version of the scheme that is a considerable improvement over one obtained via a generic lattice.

In this paper we revisit the modular lattice signature scheme and its NTRU instantiation from the following perspectives.

Security proof. In the original modular lattice signature scheme in [HPS+14], 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 [HPS+14], it requires the uSVP assumption for the NTRU lattice and the above approx-CVP assumption for the intersection of the two lattices. In an earlier version of this manuscript [HPWZ17] we introduced a new assumption named "learning with

truncation", and based the unforgeablity of the scheme on this new assumption. In this paper, we are able to give a new reduction from a standard lattice problem, namely, the Short Integer Solution (SIS) problem, under the *classical* Random Oracle Model (ROM). Prior to our work, the pqNTRUSign signature scheme did not have a formal security proof.

Sampling method. Early lattice based signature schemes, such as GGHSign [GGH97] and NTRUSign [HHP+03], 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" [NR09,DN12].

In [Lyu09], 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 [Lyu09] or a uniform distribution as in [HPS+14]). 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 [DDLL13] significantly improves the performance of a scheme.

Recall that in [HPS+14], 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 [DDLL13] and qTESLA [AAB+], 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 parametrize the scheme to be presented in this paper with the same parameters as in [DDLL13], 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 [HPS+14] 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 [DDLL13] 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 [DKL+18] 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. We note that this may make the scheme vulnerable to the attacks that exploit leakages on Gaussian samplers.

Signature aggregation. Signature aggregation allows one to compress a set of signatures into a single signature. It is often associated with batch verification, where verifying this aggregated signature implies verifying the whole set of signatures; verification operations are on the order of a single verification.

Generic aggregation/verification means the verifier is able to verify signatures signed under different public keys. A weaker notion of *same key batch verification*, on the other hand, assumes all signatures are signed under a same key. It is still a very 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 with monolithic updates) while still verifying the entire set of software images in one pass. This allows for a faster boot.

Our scheme allows for same key 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 aggregated signature is a larger vector in the lattice (compared to a single signature), we will require that the corresponding lattice problem for this aggregated signature is also hard. We will give more detail in Section 5.

We also note that schemes that realize the Fiat-Shamir heuristic, such as BLISS [DDLL13] and qTESLA [AAB$^+$], cannot provide this property easily as they need to perform the ring operations before the hash function.

Security estimations. We also give a rigorous treatment of the proposed parameters in [HWZ]. We consider best known attacks, including lattice reduction attacks [CN11,ADPS16], brute force search attacks [HS06], hybrid attacks [How07], subfield attacks [ABD16] and [HPS$^+$14].

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. 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 and a security analysis for our practical instantiation in the Section 6 and 7.

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 $\mathbf{f} := \langle f_0, f_1, \ldots, f_{n-1} \rangle$. We sometimes abuse the notation of vector and polynomial when there is no ambiguity. For a polynomial/vector $\mathbf{f}$, the norms are $\|\mathbf{f}\| := \sqrt{\sum_{i=0}^{n-1} f_i^2}$ and $\|\mathbf{f}\|_\infty := \max(|f_i|)$.

We often use the polynomial rings $\mathcal{R}_q := \mathbb{Z}[x]/F(x)$ with $F(x) = x^n + 1$. An anti-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 $\mathbf{f}_i = f(x)x^{i-1} \bmod F(x)$.

For a real a, we let $\lfloor a \rfloor$ denote the closest integer less than a. 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 and Lattices

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

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

$\mathbf{B} := (\mathbf{b}_1, \ldots, \mathbf{b}_d)^T$ 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 $\det(\mathcal{L})$ and dim denotes the determinant and dimension of the lattice $\mathcal{L}$ respectively. 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

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

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 [MP13] 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 an 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 parametrized by σ, for example, a discrete Gaussian distribution with standard deviation σ, sample uniformly at random $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \in \mathbb{Z}_q^n$; 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.

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 [Lyu12].

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 [MR07]. As suggested in [DDLL13], 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 $\mathbf{S}$ and $\mathbf{c}$ be, respectively, a fix secret matrix and vector, which consist of small norm elements, 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 [NR09,DN12] that each sample $\mathbf{x}$ leaks partial information on $\mathbf{S}$. The method used to seal this leakage is rejection sampling [Lyu09]: making the output distribution independent of $\mathbf{S}$ by probabilistically accepting the output according to certain criteria.

As shown in [Lyu12], if we wish to force the 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 [DDLL13].

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 [DDLL13] 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 [DDLL13], 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 three 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:

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

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 $\mathbf{S}_1$. By setting the parameter p such that p is co-prime to q, it can be shown that the distribution of $\mathbf{A}_2$ is statistically close to a uniform distribution by applying leftover hash Lemma when $n \approx m \log q$.

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

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

with a q-array basis $\mathbf{P} = \begin{bmatrix} 0 & q\mathbf{I}_m \\ \mathbf{I}_k & \mathbf{B} \end{bmatrix}$. The detailed algorithm that captures the above procedure is described in Algorithm 1.

Signing: The signing algorithm is described in Algorithm 2. We assume the input message digests are vectors, denoted by $\mathbf{m}_p \in \mathbb{Z}_p^n$, that are essentially the outputs of some one-way hash function. We write $\mathbf{m}_p := (\mathbf{u}_p, \mathbf{v}_p)$, with $\mathbf{u}_p \in \mathbb{Z}_p^k$ and $\mathbf{v}_p \in \mathbb{Z}_p^m$. We also use a sampler that outputs vectors from a discrete Gaussian distribution χ_σ^k.

Next, we sample a vector $(\mathbf{u}_1, \mathbf{v}_1)$ from $\mathcal{L}(\mathbf{P})$ such $\mathbf{u}_1 \equiv \mathbf{u}_p \bmod p$. To do so, one invokes the sampler to obtain $\mathbf{r} \leftarrow_\$ \chi_\sigma^k$. Then, compute $\mathbf{u}_0 = p\mathbf{r}$, $\mathbf{u}_1 = \mathbf{u}_0 + \mathbf{u}_p$, and 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 [GPV08] 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 $(\mathbf{u}_2, \mathbf{v}_2) = \mathbf{aS} = (p\mathbf{aS}_1, \mathbf{a})$. 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.

Verification: The verification process, as shown in Algorithm 3, is essentially to re-construct $\mathbf{u}$ and $\mathbf{v}$ from the signature, and then verifies the congruent condition holds.

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 needs 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 [Lyu12].

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 wraparound 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 wraparound. 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 of Ding's (R)LWE-based key exchange in 2012 [Din12], whose variants are used in [ZZD+15,Pei14,ADPS16]. We do not adopt this solution as it would increase the signature size.

Second, since the wraparound 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, as 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 $k(\log(\sigma) + 2)$ if Hoffman coding is employed. Looking ahead, we did not use this encoding since it is rather inefficient.

Algorithm 1 Key Generation Algorithm

Input: Parameters m, k, n, β, p, q.
Output: Public key $\mathbf{B}$ and secret key $\mathbf{S}_1$.
1: $\mathbf{S}_1 \leftarrow [-\beta, \beta]^{m \times k}$;
2: $\mathbf{A}_1 \leftarrow \mathbb{Z}_q^{k \times m}$;
3: $\mathbf{A}_2 = -p\mathbf{S}_1\mathbf{A}_1 \mod q$;
4: **if** $\mathbf{A}_2$ is not invertible mod q **then** go to step 2 **end if**;
5: $\mathbf{B} = \mathbf{A}_1(-\mathbf{A}_2)^{-1} \mod q$;
6: **return** $\mathbf{S}_1$, $\mathbf{B}$

3.4 Correctness

The scheme presented above is correct with probability greater than $1 - 2^{-k}$. To see that, first note that $\mathbf{b}$ is statistically close to the distribution of χ_σ^k. So for

Algorithm 2 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** $\|\mathbf{v}\|_\infty > q/2$ **then** go to step 2 **end if**
7: **return** $\mathbf{b} = (\mathbf{r} + \mathbf{a}\mathbf{S}_1)$ with probability $1/\left(M \exp\left(\frac{-2\langle \mathbf{b}, \mathbf{a}\mathbf{S}_1\rangle + \|\mathbf{a}\mathbf{S}_1\|^2}{2\sigma^2}\right)\right)$
8: go to step 2

Algorithm 3 Verification Algorithm

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

1: **if** $\|\mathbf{b}\| > 2\sigma\sqrt{k}$ **then** Reject **end if**
2: $(\mathbf{u}_p, \mathbf{v}_p) = \text{Hash}(\mu|\mathbf{B})$
3: $\mathbf{u} = p\mathbf{b} + \mathbf{u}_p$
4: $\mathbf{v} = \mathbf{u}\mathbf{B} \bmod q$
5: **if** $\mathbf{v} \not\equiv \mathbf{v}_p \bmod p$ **then** Reject **end if**
6: **return** Accept

a legitimate signature, $\|\mathbf{b}\| \leq 2\sigma\sqrt{k}$ with probability $1 - 2^{-k}$, by Lemma 3.3 of [Lyu12].

Now the verification is correct, as long as the lattice vectors remain the same for signing and verification. In addition, since the vectors are in the lattice, it is sufficient to show that the s-side of the lattice vector in the signing algorithm is equal to the s-side of the lattice vector in the verification algorithm, since the t-side vector can be deterministically generated through the s-side vector.

Note that the corresponding s-side vector in the signing algorithm is $\mathbf{u}_1 + p\mathbf{a}\mathbf{S}_1 = p(\mathbf{r}+\mathbf{a}\mathbf{S}_1)+\mathbf{u}_p$ (which has not been explicitly computed); meanwhile, in verification algorithm we have $\mathbf{u} = p\mathbf{b} + \mathbf{u}_p = p(\mathbf{r} + \mathbf{a}\mathbf{S}_1) + \mathbf{u}_p$. It follows that $\mathbf{v}$ also remains the same as one generated during the signing algorithm. Therefore, $(\mathbf{u}, \mathbf{v})$ must be the same as the corresponding ones in the signing algorithm.

What is left to shown is that $(\mathbf{u}, \mathbf{v}) \equiv (\mathbf{u}_p, \mathbf{v}_p) \bmod p$. This is straightforward from the signing algorithm.

3.5 Transcript simulation

By Theorem 4.6 of [Lyu12], the signature algorithm specified above must be negligibly close to a triple $(\mathbf{u}_p, \mathbf{v}_p, \mathbf{b})$, with distribution $U_p^k \times U_p^m \times \chi_\sigma^k$, where U_p is uniform mod p and χ_σ^k is our discrete Gaussian distribution. Such a transcript, negligibly close to a genuine transcript, can be simulated without knowledge of the secret key in the following way:

1. Choose $\mathbf{b}$ at random from χ_σ^k;
2. If $\|\mathbf{b}\| > 2\sigma\sqrt{k}$ go to step 1;
3. Set $\mathbf{u} = p\mathbf{b} + \mathbf{u}_p$, with $\mathbf{u}_p$ chosen at random mod p so that $\|\mathbf{u}\|_\infty < q/2$;
4. Set $\mathbf{v} \equiv \mathbf{uB} \mod q$, and lift $\mathbf{v}$ to the interval $(-q/2, q/2]$;
5. If $\|\mathbf{v}\|_\infty > \lfloor \frac{q}{2p} \rfloor p$ go to step 1; otherwise, output $\mathbf{v}_p \equiv \mathbf{v} \mod p$.

We claim that the distribution of the output of the actual signing algorithm is indistinguishable from that of the simulation. This is obtained from the following arguments:

- For $\mathbf{b}$: Theorem 4.6 of [Lyu12] states that the statistical distance of $\mathbf{b}$ in the simulation is within statistical distance $2^{-k}/M$ from the distribution of $\mathbf{b}$ in the actual signing algorithm.
- For $\mathbf{u}_p$: $\mathbf{u}_p$ is chosen at random from U_p^k in the transcript simulation. Therefore the distribution of $\mathbf{u}_p$ in the transcript simulation is identical to the distribution of $\mathbf{u}_p$, which is the random oracle output.
- For $\mathbf{v}_p$: First, $\mathbf{B}$ is indistinguishable from a random matrix by construction. Therefore, one can view $\mathbf{B}$ as a random mapping from $\mathbb{Z}_q^k$ to $\mathbb{Z}_q^m$. Then, since $\mathbf{u}$ has enough entropy (as $\mathbf{b}$ is sampled from χ_σ^k), $\mathbf{v}$ will be indistinguishable from random elements over $\mathbb{Z}_q^m$. Finally we use rejection sampling to ensure $\mathbf{v} \bmod p$ is uniform modulo p. With appropriate parameters, for example, $p|q$ or $q \equiv 1 \bmod p$, etc., this procedure will terminate within polynomial time.

Looking ahead, in the proof of unforgeablity, we will use a hybrid signing algorithm that resembles this simulation.

3.6 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, under the classical random oracle model. We show that our signature scheme is existential unforgeable under chosen message attacks as defined in Definition 2.

Definition 2 (EU-CMA). *A signature scheme is existential unforgeable under a chosen message attack (EU-CMA), if for any polynomial-time adversary $\mathcal{A}$, the probability of winning the following game is negligible.*

1. *The challenger $\mathcal{C}$ creates a pair of public keys pk and sk, and sends pk to the adversary $\mathcal{A}$;*
2. *$\mathcal{A}$ chooses a list of messages $m_1, \ldots, m_t$ from the message space, and receives the corresponding signatures $\sigma_1, \ldots, \sigma_t$ from the $\mathcal{C}$;*
3. *$\mathcal{A}$ produces a pair of (m', σ') where m' is not from $m_1, \ldots, m_t$;*
4. *$\mathcal{A}$ wins the game if σ' is a valid signature of m' under pk.*

In addition, the scheme is strongly unforgeable if $m' = m_i \in \{m_1, \ldots, m_t\}$, while $\sigma' \neq \sigma_i$.

Before proving the security of the scheme, we present two Lemmas which will be needed in the security reductions.

Lemma 1. *Let q be a prime. If $\mathbf{A} \in \mathbb{Z}_q^{m \times m}$ is chosen at random, then probability that it is invertible is $\frac{\prod_{i=0}^{m-1}(q^m - q^i)}{q^{m^2}}$.*

Proof. For $\mathbf{A}$ to be invertible, we need each row of $\mathbf{A}$ to be linearly independent. So the number of possible choices for the first row is $q^m - 1$. For the second row to be linearly independent, we need the second row to not be a linear combination of the first row. Thus the number of possible choices for the second row is $q^m - q$. Continuing similarly, the number of possibilities for any i-th row is $q^m - q^{i-1}$, where i indices the m rows of $\mathbf{A}$. Thus, the total number of possible matrices $\mathbf{A}$ which are invertible in $\mathbb{Z}_q$ is $\prod_{i=0}^{m-1}(q^m - q^i)$. Hence the result follows.

Lemma 2. *Let q be a composite integer which has the unique prime factorisation $q = \prod_{j=1}^{l} q_j^{e_j}$. Then the probability of a random matrix invertible in $\mathbb{Z}_q$ is $\prod_{j=1}^{l} \frac{\prod_{i=1}^{m}(q_j^m - q_j^i)}{q_j^{m^2}}$.*

Proof. First consider $j = 1$. The result follows from the fact that $\mathbf{A}$ is invertible in $\mathbb{Z}_q$ if and only if $\mathbf{A}$ is invertible in $\mathbb{Z}_{q_j}$. For $j > 1$, it follows from the Chinese Remainder Theorem.

Next, we will be using a hybrid signing algorithm in Algorithm 4 that is build from the simulation. As shown earlier, the statistical distance between the outputs of this algorithm and the actual signing algorithm is negligible.

Algorithm 4 Hybrid Signing Algorithm

Input: Message μ; Public key $\mathbf{B}$; Distribution χ_σ
Input: Parameters k, m, p, q, M
Output: A signature $\mathbf{b}$ for message μ
1: $\mathbf{b} \leftarrow \chi_\sigma^k$;
2: $\mathbf{u}_p \leftarrow \mathbb{Z}_p^k$;
3: $\mathbf{v} = (p\mathbf{b} + \mathbf{u}_p)\mathbf{B} \bmod q$;
4: **if** $\|\mathbf{v}\|_\infty > \lfloor \frac{q}{2p} \rfloor p$ **then** go to step 1 **end if**
5: $\mathbf{v}_p = \mathbf{v} \bmod p$;
6: Program the random oracle so that $(\mathbf{u}_p, \mathbf{v}_p) = \text{Hash}(\mu|\mathbf{B})$;
7: Return $\mathbf{b}$ with probability $1/M$.

Now we are ready to present the existential unforgeability result under a chosen message attack (EU-CMA) of the scheme.

Theorem 1 (Unforgeability). *Suppose there exists an adversary who queries h times to the random oracle Hash and s times to the signature algorithm, and*

can succeed in forging a signature with non negligible probability δ. Then there exists an efficient algorithm to solve the $SIS_{q,n,m,\beta}$ problem for any random matrix with probability $(1-(s+h)/p^n)\,\epsilon\delta$, where $\beta = \sqrt{16\sigma^2(n-m)+q^2m/p^2}$ and $\epsilon = \prod_{j=1}^{l} \frac{\prod_{i=1}^{m} q_j^m - q_j^i}{q_j^{m^2}}$. The running time for solving the SIS instance is the sum of the running time for the adversary's forgery, and $s+h$ times of the running time for hybrid signing algorithm.

Proof. Assume that the simulator is provided with a random matrix $\mathbf{A} := \begin{bmatrix}\mathbf{A}_1 \\ \mathbf{A}_2\end{bmatrix}$ and his goal is to find a solution to the SIS problem for $\mathbf{A}$. With overwhelming probability, $\mathbf{A}_2$ is invertible (by Lemma 2). Assume this is the case.

He first computes $\mathbf{B} = \mathbf{A}_1(-\mathbf{A}_2)^{-1}$ and publishes it as the public key for the adversary. Note that the statistical distance of the distribution of the public key in the proof and in the actual key generation algorithm is negligible, by construction.

Next, the adversary is allowed to query the random oracle (h times) and signatures (s times) of his choice. During any queries of the adversary, the simulator runs the hybrid signing algorithm (Algorithm 4) on the query input. The output will be $(\mathbf{u}_p, \mathbf{v}_p)$ for a random oracle query, and $\mathbf{b}$ for a signature query. The simulator stores $\left(\mathbf{b}^1, \mathbf{u}_p^1, \mathbf{v}_p^1\right), \left(\mathbf{b}^2, \mathbf{u}_p^2, \mathbf{v}_p^2\right), \ldots, \left(\mathbf{b}^t, \mathbf{u}_p^t, \mathbf{v}_p^t\right)$, where $t = s+h$. Note that the distinguish advantage between Algorithm 2 and Algorithm 4 is at most t/p^n, which is the probability that Algorithm 4 happens to sample a $(\mathbf{u}_p, \mathbf{v}_p)$ that the distinguisher has seen before. That is, the probability of distinguishing Algorithm 2 from Algorithm 4 is negligible in n, assuming a polynomial number of calls to the random oracle and the actual signing algorithm.

Moving ahead, the adversary will output a signature forgery $\mathbf{b}$ on a message μ of his choice. There are two scenarios. Scenario 1, the attacker does not query the random oracle on this message. In this scenario he needs to make a random guess from all possible range of the random oracle, and it's success probability is $1/p^n$ which is negligible in n.

Scenario 2, the adversary has queried the random oracle on μ. In this case, let the simulator's random oracle output for μ be $(\mathbf{u}_p^j, \mathbf{v}_p^j)$ for some $j \in [t]$. Since, $(\mathbf{u}_p^j, \mathbf{v}_p^j)$ is a random oracle output, by the construction of the hybrid signing algorithm (Algorithm 4), the simulator knows another signature $\mathbf{b}^j$ on μ, which is distinct from the attacker's except for negligible probability. This is because $\mathbf{b}^j$ was sampled at random from χ_σ^k, which contains an exponential number of candidates. From $\mathbf{b}$ and $\mathbf{b}^j$, the simulator is able to build

$$(\mathbf{u}, \mathbf{v}) := (p\mathbf{b} + \mathbf{u}_p^j, p\mathbf{c} + \mathbf{v}_p^j) \in \mathcal{L}(\mathbf{B})$$

and

$$(\mathbf{u}^j, \mathbf{v}^j) := (p\mathbf{b}^j + \mathbf{u}_p^j, p\mathbf{c}^j + \mathbf{v}_p^j) \in \mathcal{L}(\mathbf{B})$$

for some $\mathbf{c}$ and $\mathbf{c}^j$ with $\|\mathbf{c}\|_\infty$, $\|\mathbf{c}^j\|_\infty \leq q/2p$. Therefore,

$$(\mathbf{u}, \mathbf{v}) - (\mathbf{u}', \mathbf{v}') = p(\mathbf{b} - \mathbf{b}^j, \mathbf{c} - \mathbf{c}^j)$$

is also a vector in the lattice. In other words,

$$p(\mathbf{c} - \mathbf{c}^j) = p(\mathbf{b} - \mathbf{b}^j)\mathbf{B} \bmod q.$$

Also, since p is co-prime with q, we have

$$(\mathbf{b} - \mathbf{b}^j)\mathbf{A}_1 + (\mathbf{c} - \mathbf{c}^j)\mathbf{A}_2 = 0 \bmod q.$$

That is, $(\mathbf{b} - \mathbf{b}^j, \mathbf{c} - \mathbf{c}^j)$ is a solution to the SIS problem instantiated within $\mathbf{A}$.

Since $\mathbf{b}$ and $\mathbf{b}^j$ are both Gaussian vectors, we expect $\|\mathbf{b} - \mathbf{b}^j\|_2 \leq 4\sigma\sqrt{k}$. This is because ℓ_2 norm of a k dimensional Gaussian is bounded by $2\sigma\sqrt{k}$ with probability greater than $1 - 2^{-k}$ (see Lemma 3.3 of [Lyu12]). Also, by construction $\|\mathbf{c} - \mathbf{c}^j\|_\infty < q/p$, hence $\|\mathbf{c} - \mathbf{c}^j\|_2 < q/p\sqrt{m}$. To conclude, the simulator finds a short vector with ℓ_2 norm bounded by

$$\sqrt{16\sigma^2(n-m) + q^2m/p^2}.$$

Additionally, if the time required for the adversary to output a forgery is T_{Adv}, and the time required for running hybrid signing algorithm (as in Algorithm 4) is T_{Hyb}, then the simulator solves the underlying SIS problem in time $T_{Sim} \approx T_{Adv} + (s+h)T_{Hyb}$.

We remark that to ensure the provable security of SIS problem, we require 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 [AAB$^+$]). 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 new 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 [HPS$^+$14], 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 [HPS$^+$14].

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 $(1/p)^n$ 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 [GPV08].

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 which requires $n \approx m \log q$. Even if we use the ring-SIS version, the scheme is still somewhat inefficient as it requires $n \approx \log q$ - the reduction of a factor of m comes directly from the use of the ring structure in the construction. A natural way to improve its efficiency is to relax the requirement of $n \approx \log q$ (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 any more, as now $\mathbf{A}_2$ is no longer small. If we perform the same micro-adjustment as before, the coefficients of $\mathbf{v}_2$ will explode and will always cause wraparound 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: $\mathbf{H} := \mathbf{A}_1$, $\mathbf{G} := \mathbf{A}_2$ and $\mathbf{F} := \mathbf{S}_1$.

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 three 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:

$$\mathcal{L}_{\mathbf{H}} = \{(\mathbf{u}, \mathbf{v}) \in \mathbb{Z}_q^{2n} : \mathbf{uH} = \mathbf{v}\}$$

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 a few differences.

First, we use rejection samplings on keys. This gives us better bounds for $\|\mathbf{af}\|_\infty$ and $\|\mathbf{ag}\|_\infty$ during signing, and in return ensures a better rejection rate. Note that key rejection isn't uncommon. For example, it was also performed in BLISS [DDLL13], with a different rejection method.

Second, 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.

Third, 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 [HPS+14], after rejection sampling, the distribution of $\mathbf{v}$ will be computationally indistinguishable from uniform over $(-\frac{q}{2} + B_t, \frac{q}{2} - B_t)$ for a public parameter B_t which depends on q, the (uniform) distribution of $\mathbf{a}$, and the number of $+1$s and -1s in $\mathbf{g}$.

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 5. We use the classical NTRU flat form (non-product form, cf. [HPS+17]) 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 $f(1) = g(1) = 1$.

Now we give arguments for rejection sampling on $\mathbf{f}$. A similar argument applies to $\mathbf{g}$. We wish to estimate $\|\mathbf{af}\|_\infty$ where $\mathbf{a}$ is assumed a uniform binary vector. We view $\mathbf{af}$ as a vector-matrix multiplication: $\mathbf{aF}$. Since $\mathbf{a}$ is binary, this operation can be seen as randomly adding some coefficients from each row of $\mathbf{F}$. Therefore for a given $\mathbf{f}$, the corresponding $\|\bar{\mathbf{f}}\|_\infty$ will correlate with the

Algorithm 5 Key Generation Algorithm

Input: Parameters N, p, q, d, B_k.
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: $\mathbf{F} \leftarrow$ Cyclic matrix of $\mathbf{f}$, and $\bar{\mathbf{f}} \leftarrow$ sum of rows of $\mathbf{F}$;
4: **if** $\|\bar{\mathbf{f}}\|_\infty > B_k$ **then** go to step 1 **end if**;
5: $\mathbf{g} \leftarrow T(d+1, d)$;
6: **if** $\mathbf{g}$ is not invertible mod p **then** go to step 5 **end if**
7: $\mathbf{G} \leftarrow$ Cyclic matrix of $\mathbf{g}$, and $\bar{\mathbf{g}} \leftarrow$ sum of rows of $\mathbf{G}$;
8: **if** $\|\bar{\mathbf{g}}\|_\infty > B_k$ **then** go to step 5 **end if**;
9: $\mathbf{h} = \mathbf{g}/(p\mathbf{f}) \bmod q$
10: **return** $\mathbf{h}$, $\mathbf{g}$ and $\mathbf{f}$

expected value of $\|\mathbf{af}\|_\infty$. Hence, rejection sampling based on $\|\bar{\mathbf{f}}\|_\infty$ will ensure that $\|\mathbf{af}\|_\infty$ is also small. Note that with the parameters given in this paper, our key space after the rejection sampling is still large enough to be robust against search attacks.

Algorithm 6 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_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: $\mathbf{a} = (\mathbf{v}_p - \mathbf{v}_1)/\mathbf{g} \bmod p$ {Additional $\mathbf{g}^{-1}$ factor}
5: $\mathbf{v} = \mathbf{v}_1 + (-1)^b\mathbf{ag}$;
6: **if** $\|\mathbf{v}\|_\infty > q/2 - B_t$ **then** go to step 2 **end if** {Additional rejection sampling on t side}
7: **return** $\mathbf{b} = (\mathbf{r} + (-1)^b\mathbf{af})$ with probability $1/\left(M_s \exp\left(-\frac{\|\mathbf{af}\|}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{b}, \mathbf{af}\rangle}{\sigma^2}\right)\right)$
8: go to step 2

Signing algorithm : The signing algorithm is shown in Algorithm 6. We highlight the differences between this signing algorithm and the one described in previous section.

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

Second, in step 6, rejection sampling is performed on the t-side, parametrized by an additional integer B_t. The distribution of the t-side vector will be uniform within the interval $(-\frac{q}{2} + B_t, \frac{q}{2} - B_t)$. The public parameter B_t is also computed as an average over a large number of choices of $\mathbf{g}$ and $\mathbf{a}$. Since we have performed

rejection samplings on $\mathbf{g}$ during the sign algorithm, we obtain a better bound for $\mathbf{ag}$ which reduces the number of repetitions.

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 [DDLL13], 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 7 Verification Algorithm

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

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

Verification : The verification algorithm in shown in Algorithm 7. It firstly checks the signature is in the correct range. It then re-compute the lattice vector, and checks if the vector meets both length requirement, and the congruent condition.

4.3 Transcript simulation

As in Section 3.5, a transcript, indistinguishable from a genuine transcript, can be simulated without knowledge of the secret key in the following way. Note that the only difference here is $p|(q/2 - B_t)$ in our setting.

1. Choose $\mathbf{b}$ at random from χ_σ^N;
2. Reject if $\|\mathbf{b}\| > 2\sigma\sqrt{N}$ and return to Step 1.
3. Set $\mathbf{u} = p\mathbf{b} + \mathbf{u}_p$, with entries of $\mathbf{u}_p$ chosen at random mod p;
4. Set $\mathbf{v} \equiv \mathbf{uh} \bmod q$, and lift $\mathbf{v}$ to the interval $[-q/2, q/2)$;
5. Reject if $\|\mathbf{v}\|_\infty > q/2 - B_t$ and return to Step 1. Otherwise set $\mathbf{v}_p \equiv \mathbf{v} \bmod p$ and accept $\mathbf{b}$ as a signature on $\mathbf{u}_p, \mathbf{v}_p$.

4.4 Security

We may apply a same proof technique as in the SIS-based construction in Section 3.6. However, a challenger will only be able to recover a lattice vector $(\mathbf{b}', \mathbf{c}') \in \mathcal{L}_h$, that, although is somewhat short, remains significantly larger than the secret keys $\mathbf{f}$ and $\mathbf{g}$. Therefore, with a same reduction, one is able to show that a forger against EU-CMA security implies a solver of the SIS problem over the NTRU lattice $\mathcal{L}_h$; although the precise hardness of this problem is not established. We omit the details of the proof since it will be identical to the previous one.

5 Batch verification

The modular lattice signature scheme presented here allows for same key 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 wraparound 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 wraparound mod q.

To be precise, suppose we want to verify k signatures in one batch, corresponding to $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(k)}$ vectors on the s-side. We require that $\| \sum_{i=0}^{k} \mathbf{u}^{(i)} \|_\infty < q/2$ to not cause a wraparound. Since $\mathbf{u} = p\mathbf{b} + \mathbf{u}_p$, this is effectively the sum of

1. k samples from discrete Gaussian, multiplied by p, and
2. k samples from uniform distribution modulo p.

We approximate the first quantity by replacing discrete Gaussian with continuous Gaussian. Then, the sum of k samples will be close to samples from continuous Gaussian with variance $\sqrt{k}\sigma$ except for negligible probability. Use the tail-cutting property, we know that the first quantity is bounded by $p\tau\sqrt{k}\sigma$. Next, for the second quantity, we know that the infinity norm for a sum of k binary vectors of dimension N will be bounded by k.

Finally, we conclude that our scheme can batch verify k signatures so long as

$$p\tau\sqrt{k}\sigma + k < q/2.$$

For our parameter choices, to be shown in Section 6, we have $k = 24$, $q = 2^{16} + 1$, $\sigma = 250$ and $\tau = 13.3$. See Algorithm 8, below, for the batch verification algorithm.

Algorithm 8 Batch Verification Algorithm

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

1: $(\mathbf{u}_{p,i}, \mathbf{v}_{p,i}) = \text{Hash}(\mu_i|\mathbf{h})$
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 Known attacks on 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 $1 \le i \le k$, do the following:

- for each $\mathbf{v}_p^{(i)}$, find a vector $\mathbf{v}_1^{(i)}$ such that $\mathbf{v}_1^{(i)} = \mathbf{v}_p^{(i)} \bmod p$; this can be done by adding some vector, that is a multiple of p, to $\mathbf{v}_p^{(i)}$;
- 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 $\mathbf{v}_1^{(i)}$'s such that $\mathbf{v}_1^{(i)}$ is congruent to $\mathbf{v}_p^{(i)}$ for each message mod p. In the meantime, the $\mathbf{U}$ vector also meets 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).

5.2 Security

We define the unforgeability of the aggregated signature scheme through the following game:

1. The challenger $\mathcal{C}$ creates a pair of public keys pk and sk, and sends pk to the adversary $\mathcal{A}$;
2. $\mathcal{A}$ chooses a list of messages $m_1, \ldots, m_t$ from the message space, and receives the corresponding signatures $\sigma_1, \ldots, \sigma_t$ from the $\mathcal{C}$;
3. $\mathcal{A}$ receives another list of message and signature pairs $m^{(1)}, \ldots, m^{(k-1)}$ and $\sigma^{(1)}, \ldots, \sigma^{(k-1)}$. Those $m^{(i)}$ may be from $\{m_1, \ldots, m_t\}$;
4. $\mathcal{A}$ produces a pair of $(m^{(k)}, \sigma)$ where $m^{(k)}$ is not from $m_1, \ldots, m_t$;
5. $\mathcal{A}$ wins the game if σ is a valid signature of $m^{(1)}, \ldots, m^{(k)}$ under pk.

We sketch the proof here. There are two ways that a forger may succeed in this game. First scenario, the forger is able to generate a valid signature $\sigma^{(k)}$ for $m^{(k)}$; it follows that the forger can easily build the aggregated signature since all k signatures are valid. The success probability of the forger is negligible in this setting, since the underlying signature algorithm is EU-CMA secure.

Alternatively, in the second scenario, the forger may try to forge a σ for which it does not know $\sigma^{(k)}$. In this case, the forger has to forge a $\sigma = \mathbf{V}$ for some random $\mathbf{u}_p, \mathbf{v}_p$, where $(\mathbf{u}_p, \mathbf{v}_p) = \sum_{i=1}^{k} \left(\mathbf{u}_p^{(k)}, \mathbf{v}_p^{(k)}\right)$. Then, the simulator will be able to compute $\mathbf{U} = \mathbf{V}\mathbf{h}^{-1}$. That is,

$$\left(p\mathbf{b} + \sum_{i=1}^{k} \mathbf{u}_p^{(i)}\right)\mathbf{h} = p\mathbf{c} + \sum_{i=1}^{k} \mathbf{v}_p^{(i)}$$

for some $\mathbf{c}$.

In the meantime, since the hash function is modelled as a random oracle and the forger doesn't know the hash value of $m^{(k)}$, similar to the proof for the fundamental game, the simulator will have a list of tuples $\left(\mathbf{u}_p^{(k)}, \mathbf{v}_p^{(k)}, \mathbf{b}^{(k)}\right)$. From this list, the simulator is also able to get

$$\left(p\mathbf{b}' + \sum_{i=1}^{k} \mathbf{u}_p^{(i)}\right)\mathbf{h} = p\mathbf{c}' + \sum_{i=1}^{k} \mathbf{v}_p^{(i)}$$

for $\mathbf{b}' = \sum_{i=1}^{k} \mathbf{b}^{(i)}$, and some $\mathbf{c}'$. Therefore, the simulator will be able to extract the same solution $(\mathbf{b} - \mathbf{b}', \mathbf{c} - \mathbf{c}')$ to the SIS problem as in the previous proof.

Note that a major difference here is on the quality of the extracted vector. In this setting, the extracted vector will be larger (it will be k times larger compared to the extracted vector from the basic signature scheme). Therefore, we need to increase the parameters of the scheme so that even extracting a larger vector is hard based on best known attacks. This also confirms our observation in the previous subsection.

6 Parameters and benchmarks

We give 2 sets of parameters. The first one, namely, Gaussian-1024, is designed in accordance with the scheme described in Section 4. For completeness, we also provide parameters for the original pqNTRUSign scheme. Its parameter set is named Uniform-1024. The pqNTRUSign signature scheme, with both parameters, was submitted to NIST for standardization consideration for post-quantum cryptography.

We estimate that Gaussian-1024 and Uniform-1024 provide 269 bits of classical security and 149 bits of quantum security. The details of the above estimations shall be presented in Section 7. Note that the parameter sets in this proposal are not the same as in [HPS$^+$14,HPWZ17] due to the use of a different estimation model. In previous estimations [HPS$^+$14,HPWZ17], the authors followed the work of [CN11] and [DDLL13] by assuming that a root Hermite factor of 1.005 is not achievable. Here we have used a more rigorous estimation for our new parameter sets. To be a bit more detailed, we used BKZ with core-sieving model to estimate the cost of attacks to be presented in the next section, for both classical and quantum security, respectively.

Table 1. Parameters.

Param	$\mathcal{R}, N, q$	d_f, d_g	σ	B_k, B_s, B_t	Raw PK size	Raw Sig size
Gaussian-1024	$\frac{\mathbb{Z}_q[x]}{x^N+1}, 1024, 2^{16}+1$	205	250	40, 500,49	16384 bits	$\approx$ 11264 bits
Uniform-1024	$\frac{\mathbb{Z}_q[x]}{x^N+1}, 1024, 2^{16}+1$	205	N/A	40, 98, 49	16384 bits	16384 bits

We implemented our scheme with C. Our software is available at [git] 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.

Table 2. Benchmark results

	Gaussian-1024	Uniform-1024
Key Generation	47.8 ms / 124 280 k cycles	48.9 ms / 127 140 k cycles
Signing	120 ms / 312 000 k cycles	72 ms / 187 200 k cycles
Verification	0.96 ms / 2 496 k cycles	0.97 ms / 2 522 k cycles
Batch Verification (10 signatures)	1.01 ms / 2 626 k cycles	1.02 ms / 2 649 k cycles

We present the implementation results in Table 2. 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.

6.1 Comparison

We give a brief comparison between our scheme and the other candidates in the NIST PQC standardization process in Table 3. Our scheme is significantly slower. Note that the benchmark for Dilithium and qTesla are based on their AVX-2 implementations. Our implementation is significantly slower mainly due to a slower ring multiplication implementation, as well as a much larger rejection sampling parameter. We also admit that due to this large rejection sampling parameter, it is unlikely that our scheme will outperform any of the three even if we have an AVX-2 accelerated implementation for ring multiplication.

Table 3. Comparison with other candidates

	Dilithium 1024×768	qTESLA-III	Falcon 1024	Uniform-1024
PK (bytes)	1184	3104	1793	2048
Signature (bytes)	2044	2848	1234	2048
Key gen (ms)	0.03	0.85	19.64	48.9
signing (ms)	0.11	0.15	0.32	72
verification (ms)	0.04	0.05	0.06	0.97

7 Best known attacks

7.1 Overview

In this evaluation, we will

1. follow the new analysis in [ADPS16] using BKZ 2.0 with classical sieving to estimate the **classical security**.
2. follow the new analysis in [ADPS16] using BKZ 2.0 with quantum sieving to estimate the **quantum security**.

For completeness, we also give the analysis result of

3. the original BKZ 2.0 analysis [CN11] with the extreme pruning method;
4. the (quantum) hybrid attacks following the sequence of work [How07,Wun16,GvVW17,Wun19].

Table 4. Lattice strength given by root Hermite factor

	Gaussian-1024	Uniform-1024
Public key strength $\left(\frac{\mathrm{GH}}{\lambda_1}\right)^{\frac{1}{2N}}$	$524^{\frac{1}{2048}} \approx 1.0030$	
Forgery strength $\left(\frac{\|\mathbf{u},\mathbf{v}\|}{\mathrm{GH}}\right)^{\frac{1}{2N}}$	$25^{\frac{1}{2048}} \approx 1.0016$	$153^{\frac{1}{2048}} \approx 1.0024$

For signatures, there are two types of attacks, namely, public key attacks which try to recover the secret key from a public key, and forgery attacks, which try to forge a signature without a secret key. As we shall see from Table 4, forgery attacks are strictly harder than public key attacks, due to the smaller root Hermite factors. Therefore, our focus here is to evaluate the security of the public keys, i.e., recovering $\mathbf{f}$ or $\mathbf{g}$ from $\mathbf{h}$.

Here, we summarize the evaluations in Table 5. We estimate a 165 bits security in the classical setting, and a 149 bits security in the quantum setting.

Table 5. Best known attacks and their bit complexity on public keys

N	BKZ + Enum		BKZ + Sieving		BKZ + QSieving	
	uSVP	Hybrid	uSVP	Hybrid	uSVP	Hybrid
1024	407	269	**165**	**165**	**149**	154

7.2 Lattice attacks on public keys

For an NTRU public key polynomial $\mathbf{h}$, let $\mathbf{H}$ be the matrix whose row vectors are the cyclic rotation of $\mathbf{h}$. Then the NTRU lattice associated with $\mathbf{h}$ uses a basis of the rows of

$$\mathbf{B} = \begin{bmatrix} q\mathbf{I}_N & 0 \\ \mathbf{H} & \mathbf{I}_N \end{bmatrix}$$

where $\mathbf{I}_N$ is an N-dimensional identity matrix. Within this NTRU lattice, there exist unique shortest vectors, namely, the vector form of $\langle \mathbf{f}, \mathbf{g} \rangle$ and its cyclic rotations.

This attack was first presented in the original NTRU paper [HPS98] circulated during the rump session of Crypto'96. It was later observed in [CS97] that one does not necessarily need to find the exact secret key to be able to decrypt. An attack is successful if the attacker can locate any vectors in this lattice that are sufficiently small (such as a cyclic rotation of the secret key).

It has been shown in [GN08] that the ability to locate a unique shortest vector in a lattice depends on the root Hermite factor of the lattice, which is the n-th root of

$$\frac{\text{Gaussian expected length}}{l_2 \text{ norm of the target vector}}$$

where n is the dimension of the lattice.

Here, we give an estimate for the root Hermite factor corresponding to the proposed parameter set. This lattice has a dimension of $2N$. The Gaussian expected length of the shortest vector in this lattice is

$$\sqrt{qN/\pi e},$$

while the l_2 norm of the target vectors are $\|\mathbf{f}, \mathbf{g}\|_2$. This gives the root Hermite factor of the lattice as

$$\left(\frac{\sqrt{Nq/\pi e}}{\|\mathbf{f}, \mathbf{g}\|_2}\right)^{\frac{1}{2N}}.$$

It was originally believed that the current technique of BKZ 2.0 [CN11] is only able to find a short vector with a root Hermite factor of at least 1.005. However, in [ADPS16], the authors give a conservative analysis of the cost of BKZ 2.0 reduction. As pointed out by the authors themselves, those estimations are very optimistic about the abilities of an attacker. In particular, unlike the analysis of BKZ 2.0 [CN11], where the cost of shortest vector subroutines is estimated via the cost of enumeration with extremely pruning [GNR10], this analysis assumes that for large dimensional lattices, shortest vector problems can be solved very efficiently using heuristic sieving algorithms, ignoring the sub-exponential to exponential requirement of space.

Giving a few details, the best known classical and quantum sieving algorithms have time costs of $2^{0.292n}$ and $2^{0.265n}$, respectively [BLS16]. The best plausible quantum short vector problem solver costs more than $2^{0.2075n}$ since this is the space required to store the list of vectors. In practice, sieving tends to process much slower than enumeration techniques. Moreover, sieving algorithms require a similar level of space complexity (exponential in n), while the space requirement of enumeration techniques is polynomial.

For the sake of completeness, we present the estimated cost of BKZ with classical and quantum sieving algorithms, following the methodology of [ADPS16]. It is easy to see that the space requirement for classical sieving algorithms is far from practical. For example, it is estimated that the world's storage capacity is around 295 exabytes $\approx 2^{68}$ bits [sto11]; and the number of atoms in the whole earth is around $10^{49} \approx 2^{162}$ [ato14]. Thus we do not use BKZ with classical sieving to estimate the classical security of our parameters. Nonetheless, we do

use BKZ with quantum sieving algorithms to estimate the quantum security, in accounting for unknown effects on data storages with quantum computers.

7.3 Search attack

For NTRU with trinary keys, since the secret keys are trinary polynomials with df number of 1s and -1s, the search space for the secret key is $\binom{N}{df+1,df}/N$. For example, with our parameter set, we have 2^{1384} candidates. (The factor $1/N$ comes from the fact that an attacker can guess any of N anti-cyclic rotations of the secret key, rather than just the secret key itself.) We remark that this key space for our parameter set is considerably larger than that in [HPS+17] due to the switch from product form polynomials to flat form polynomials. This is sufficient even with the presence of meet-in-the-middle attacks [HS06] and quantum attacks using Grover's algorithm [Gro96].

7.4 Hybrid attack

The previous best known attack against NTRU, prior to the BKZ with quantum sieving analysis [ADPS16], was the hybrid attack [How07] which is a hybrid of a lattice attack and a meet-in-the-middle search attack. It has been revisited recently, by a sequence of work [Wun16,GvVW17,Wun19]. In this paper, we use the hybrid estimator from [Sch] to obtain the cost for hybrid attacks. We present the cost of the classical hybrid attack and compare it with solving directly the uSVP in Table 5.

7.5 Subfield attack

Subfield attacks against NTRU have been considered in [Ber14]. It was reported in [ABD16] that for certain "over-stretched" NTRU parameters, one can exploit a subfield. This attack was only applicable to the NTRU lattices that are used to instantiate a (fully) homomorphic encryption scheme. The authors of [ABD16] also showed that for our parameters the subfield attack will not be successful.

7.6 Forgery attack

The best forgery attack, other than deriving the secret keys from the public keys via the above attacks, is the lattice reduction attack shown in [HPS+14]. 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.

For Uniform-1024, each coordinate of $\mathbf{u}$ (and $\mathbf{v}$) is approximately randomly and uniformly distributed between $-q/2+B_s$ and $q/2-B_s$ ($-q/2+B_t$ and $q/2-B_t$, resp.) . Ignoring the B_s, the average squared coefficient will be approximately

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

Thus $\mathbf{u}$ and $\mathbf{v}$ will have norm $\|\mathbf{u}\|^2 \approx \|\mathbf{v}\|^2 \approx q^2N/12$. We thus have

$$\frac{\text{Target Length}}{\text{Gaussian Heuristic Length}} = \frac{\sqrt{q^2N/6}}{\sqrt{Np^2q/\pi e}} = \sqrt{\frac{q\pi e}{6p^2}}.$$

For Gaussian-1024, 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 the lattice generated by 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 chooses 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.

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

8 Conclusion

In this paper we revisit the modular lattice signature scheme, and its instantiation using NTRU lattices. The main improvement is a proof of the strong unforgeability of the signature based on the short integer solution problem over the corresponding lattice. In terms of practice, our non-optimized, non-factorized implementation is not as efficient as other solutions in this domain, such as Dilithium and Falcon. We leave this to future work.

References

AAB+. Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Nina Bindel, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Krämer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. qTESLA: Efficient and Post-quantum secure lattice-based signature scheme. `https://qtesla.org`.

ABD16. Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 153–178, 2016.

ADPS16. 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.

ato14. The number of atoms in the World, 2014. available from `http://www.fnal.gov/pub/science/inquiring/questions/atoms.html`.

Ber14. Daniel J. Bernstein. A subfield-logarithm attack against ideal lattices, 2014. available from `https://blog.cr.yp.to/20140213-ideal.html`.

BLS16. Shi Bai, Thijs Laarhoven, and Damien Stehlé. Tuple lattice sieving. *IACR Cryptology ePrint Archive*, 2016:713, 2016.

CJL+16. 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.

CN11. Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 1–20, 2011.

CS97. Don Coppersmith and Adi Shamir. Lattice attacks on NTRU. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pages 52–61, 1997.

DDLL13. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *CRYPTO 2013*, pages 40–56, 2013.

Din12. Jintai Ding. A simple provably secure key exchange scheme based on the learning with errors problem. *IACR Cryptology ePrint Archive*, 2012:688, 2012.

DKL+18. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.

DLP14. 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.

DN12. 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.

FHK+. Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William

Whyte, and Zhenfei Zhang. Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. `https://falcon-sign.info/`.

GGH97. 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.

git. NTRU OpenSource Project. online. available from `https://github.com/NTRUOpenSourceProject/ntru-crypto`.

GN08. Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, pages 31–51, 2008.

GNR10. Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, pages 257–278, 2010.

GPV08. 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 197–206, New York, NY, USA, 2008. ACM.

Gro96. Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219, 1996.

GvVW17. Florian Göpfert, Christine van Vredendaal, and Thomas Wunderer. A hybrid lattice basis reduction and quantum search attack on LWE. In *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*, pages 184–202, 2017.

HHP+03. 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.

How07. Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, pages 150–169, 2007.

HPS98. 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.

HPS+14. Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, and William Whyte. Transcript secure signatures based on modular lattices. In *PQCrypto 2014*, pages 142–159, 2014.

HPS+17. Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for ntruencrypt. In *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings*, pages 3–18, 2017.

HPWZ17. Jeffrey Hoffstein, Jill Pipher, William Whyte, and Zhenfei Zhang. A signature scheme from learning with truncation. Cryptology ePrint Archive, Report 2017/995, 2017. https://eprint.iacr.org/2017/995.

HS06. Jeffrey Hoffstein and Joseph H. Silverman. Meet-in-the-middle Attack on an NTRU private key, 2006. available from http://www.ntru.com.

HWZ. Cong Chen Jeffrey Hoffstein, William Whyte, and Zhenfei Zhang. pqNTRUSign.

Lyu09. Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT 2009*, page 598–616. Springer, 2009.

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. 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 738–755, 2012.

MP13. 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.

MR07. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

NIS. NIST. Post-Quantum Cryptography - Round 1 Submissions. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions.

NR09. Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *J. Cryptology*, 22(2):139–160, 2009.

NSA. NSA Suite B Cryptography - NSA/CSS.

Pei14. 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.

Sch. John Schanck. Estimator: Scripts for estimating the security of lattice based cryptosystems. https://github.com/jschanck/estimator.

Sho94. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 124–134, 1994.

sto11. What is the world's data storage capacity?, 2011. available from http://www.zdnet.com/article/what-is-the-worlds-data-storage-capacity/.

Wun16. Thomas Wunderer. Revisiting the hybrid attack: Improved analysis and refined security estimates. *IACR Cryptology ePrint Archive*, 2016:733, 2016.

Wun19. Thomas Wunderer. A detailed analysis of the hybrid lattice-reduction and meet-in-the-middle attack. *J. Mathematical Cryptology*, 13(1):1–26, 2019.

ZZD+15. Jiang Zhang, Zhenfeng Zhang, Jintai Ding, Michael Snook, and Özgür Dagdelen. Authenticated key exchange from ideal lattices. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 719–751, 2015.