Compact Lattice Signatures via Iterative Rejection Sampling

Joel Gärtner[1,2][0000−0002−3724−2914]

[1] KTH Royal Institute of Technology, Stockholm, Sweden jgartner@kth.se
[2] Swedish NCSA, Swedish Armed Forces, Stockholm, Sweden

Abstract. One of the primary approaches for constructing lattice-based signature schemes is through the "Fiat-Shamir with aborts" methodology. Schemes constructed using this approach may abort and restart during signing, corresponding to rejection sampling produced signatures in order to ensure that they follow a distribution that is independent of the secret key. This rejection sampling is only feasible when the output distribution is sufficiently wide, limiting how compact this type of signature schemes can be.

In this work, we develop a new method to construct lattice signatures with the "Fiat-Shamir with aborts" approach. By constructing signatures in a way that is influenced by the rejection condition, we can significantly lower the rejection probability. This allows our scheme to use an iterative rejection sampling to target narrower output distributions than previous methods, resulting in much more compact signatures.

In the most compact variant of our new signature scheme, the combined size of a signature and a verification key is less than half of that for ML-DSA and comparable to that of compact hash-and-sign lattice signature schemes, such as Falcon. Alternatively, by targeting a somewhat wider distribution, the rejection condition of the scheme can be securely ignored. This non-aborting variant of our scheme still retains a notable size advantage over previous lattice-based Fiat-Shamir schemes.

1 Introduction

As a part of their post-quantum standardization process, NIST has already standardized the signature scheme CRYSTALS-Dilithium [26] as ML-DSA and is in the process of standardizing Falcon. Both Dilithium and Falcon are lattice-based signature schemes but they are constructed using two different approaches, corresponding to the two main approaches for constructing lattice-based signature schemes. The first of these is the hash-and-sign approach of Gentry, Peikert and Vaikuntanathan [21] on which Falcon is based. The second of these, on which Dilithium is based, is the one of Lyubashevsky [24, 25], which consists of an aborting variant of the Fiat-Shamir transform.

One of the primary reasons for why NIST decided to standardize Falcon in addition to Dilithium is that Falcon has significantly smaller signatures and verification keys. In fact, to the best of our knowledge, there has

not previously been any lattice-based Fiat-Shamir signature scheme as compact as Falcon.

A limiting factor in constructing a compact Fiat-Shamir lattice-based signature scheme is that, for its security proof, it is typically required that it produces signatures that follow some distribution which is independent of the scheme's secret. To accomplish this using the Fiat-Shamir with aborts approach, the scheme initially produces candidate signatures from a distribution that is dependent on the secret of the scheme. Only some of these signatures are actually emitted, with the scheme otherwise aborting the signing attempt and retrying. This corresponds to rejection sampling from a secret-dependent distribution to produce signatures that follow a secret-independent distribution. The rejection probability of this sampling is determined by the ratio between the size of the secret and the width of the output distribution. Because of this, unless the output distribution is sufficiently wide in relation to the size of the secret, the rejection probability is too high for it to be useable.

Altering how signatures are constructed can allow for a narrower output distribution, and thus smaller signatures, while keeping the rejection probability the same. Furthermore, the estimated security of these types of signature schemes is in large part determined by the maximal size of accepted signatures and if the scheme produces smaller signatures, we can argue for a higher security level. Alternatively, the smaller signatures allow many other parameters of the scheme to be selected differently when targeting a comparable security level. As such, an alternative signature construction that allows for better rejection sampling can influence many different aspects of the scheme.

BLISS [16] provides an example of the impact of improved rejection sampling, where rejection sampling from a bimodal Gaussian distribution allows for a significantly more compact scheme than Lyubashevsky's original approach. A more recent example is HAETAE [11, 10] which relies on bimodal rejection sampling from uniform distributions over hyperballs to allow for a more compact scheme than Dilithium. It has even been shown in [13] that the approaches of BLISS and HAETAE are essentially optimal for schemes that rely on rejection sampling from bimodal distributions, and that rejection sampling from bimodal distributions is superior to similar sampling from unimodal distributions.

Very recently, the NTRU+Sign [30] signature scheme was introduced with various improvements to the BLISS signature scheme. This same work also recalibrated the parameters of the original BLISS scheme in order to ensure alignment with modern security standards.

Works have also explored the possibility of removing the rejection condition [7]. A concrete example of this is Raccoon [12], which was a submission to the first round of NIST's post-quantum standardization process for additional digital signature schemes. The Raccoon parameters are selected to ensure that rejection conditions can be securely ignored. This does, however, come at the cost of signatures that are about five times larger than those of Dilithium.

An alternative approach for constructing signatures without a rejection condition was developed for the G+G signature scheme [14]. Signatures from this scheme follow a discrete Gaussian distribution that is con-

structed as a convolution of two different discrete Gaussian distributions, both of which have covariance that depend on the secret key of the scheme. By carefully selecting parameters, the signatures follow a centered spherical discrete Gaussian distribution, ensuring that the signatures do not leak anything about the secret.
Even though the G+G scheme is constructed without relying on aborts, the resulting scheme is still relatively compact. The G+G scheme is, however, still far from as compact as Falcon. Additionally, due to requiring sampling from Gaussian distributions dependent on the secret key, it seems like implementing G+G securely would be non-trivial.

1.1 Our Contribution

In this work we present an alternative method for constructing lattice-based signatures for the Fiat-Shamir with aborts paradigm. By constructing signatures in a way that is influenced by the rejection condition, our new method allows for a rejection probability that is significantly smaller than that of previous methods. A lower rejection probability allows for greater freedom when parametrizing the scheme, allowing us to construct a signature scheme that is significantly more compact than previous lattice-based Fiat-Shamir signature schemes.
We propose concrete parametrization of a signature scheme that makes use of our new method. The signature size of the resulting scheme is approximately one third of that for ML-DSA, and the verification key size is somewhat smaller than that for ML-DSA. Additionally, the combined size of a signature and a verification key is less than half of what it is for ML-DSA and within 10% to that of Falcon.
We also consider alternative parametrization of our scheme where the rejection condition can be safely ignored. In a similar manner as for Raccoon [12], this is accomplished by using a wider output distribution. However, the penalty that we pay, in the form of increased size, is significantly smaller than for Raccoon, and our resulting scheme without aborts is still more compact than previous lattice-based Fiat-Shamir signature schemes. In particular, the combined size of a signature and a verification key for the resulting scheme is less than 60% of that for ML-DSA.

1.2 Technical overview

The idea behind Lyubashevsky's signature scheme [24, 25] is similar to the idea behind Schnorr signatures [29]. To sign a message with Lyubashevsky's scheme, a vector $\boldsymbol{y}$ is sampled from some relatively narrow distribution. Then, a commitment $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{y} \bmod q$ is computed, where $\boldsymbol{A}$ is a matrix that is part of the verification key and q is a parameter for the scheme.
Using the commitment $\boldsymbol{w}$ and the message to be signed, a challenge $\boldsymbol{c}$ is derived, with this challenge guaranteed to be a short vector. A signature is given by $(\boldsymbol{z}, \boldsymbol{c}, \boldsymbol{w})$, where $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{S}\boldsymbol{c} \bmod q$ with $\boldsymbol{S}$ the secret key of the signer. The secret key $\boldsymbol{S}$ is sampled to have short columns, and thus $\boldsymbol{S}\boldsymbol{c}$ is relatively short, ensuring that $\boldsymbol{z}$ also is a relatively short vector.

In contrast to the analogous case for typical Schnorr signatures, in the lattice setting the vector $\boldsymbol{y}$ does not perfectly mask the contribution of $\boldsymbol{Sc}$ to $\boldsymbol{z}$. This is the case as $\boldsymbol{z}$, and therefore also $\boldsymbol{y}$, must be relatively short vectors in order to ensure that signatures are hard to forge. As such, the distribution of signatures produced in this manner depend on $\boldsymbol{S}$ and may therefore leak information about the secret key of the scheme.
To ensure that the distribution of signatures does not depend on secret information, Lyubashevsky [24, 25] does not output all constructed signatures. Instead, signatures are only emitted with some probability that depends on $\boldsymbol{z}$, and otherwise the signing attempt is aborted and retried with a different $\boldsymbol{y}$. By aborting with a suitable probability, it is ensured that the $\boldsymbol{z}$ vector in emitted signatures follow some secret-independent distribution. As such, emitted signatures do not leak any information about $\boldsymbol{S}$.
A variant of Lyubashevsky's scheme was later considered for the signature scheme BLISS [16]. In BLISS, valid signatures can be constructed with $\boldsymbol{z}$ either as $\boldsymbol{y} + \boldsymbol{Sc}$ or as $\boldsymbol{y} - \boldsymbol{Sc}$, with each of these two constructions for $\boldsymbol{z}$ selected with probability $1/2$. As with Lyubashevsky's original scheme, BLISS occasionally aborts and does not output the constructed $\boldsymbol{z}$.
As $\boldsymbol{y}$ is sampled from a discrete Gaussian distribution in BLISS, without rejection sampling, the distribution of BLISS signatures follows a bimodal discrete Gaussian distribution, with the two centers $\pm\boldsymbol{Sc}$. This is in contrast to the scheme of Lyubashevsky [25] where the distribution of signatures would be a unimodal Gaussian distribution with center $\boldsymbol{Sc}$ if no rejection sampling was used.
In both the unimodal and bimodal cases, a suitable rejection condition allows the output signatures to follow a centered discrete Gaussian distribution. However, the bimodal discrete Gaussian distribution is significantly more similar to the desired output distribution than a unimodal discrete Gaussian distribution centered on $\boldsymbol{Sc}$. This gives the bimodal approach of BLISS a significant advantage over the original unimodal approach of Lyubashevsky.

Our Improved Signature Construction The rejection sampling in BLISS can be described through a rejection function $R(\boldsymbol{z})$, with each constructed $\boldsymbol{z}$ accepted with probability $R(\boldsymbol{z})$ and otherwise rejected. For a sampled vector $\boldsymbol{y}$, the signature is given by $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{Sc}$ with probability $R(\boldsymbol{y} + \boldsymbol{Sc})/2 = f_{\boldsymbol{Sc}}(\boldsymbol{y})$ and by $\boldsymbol{z} = \boldsymbol{y} - \boldsymbol{Sc}$ with probability $R(\boldsymbol{y} - \boldsymbol{Sc})/2 = g_{\boldsymbol{Sc}}(\boldsymbol{y})$, while it otherwise rejects the signature and tries again with a different $\boldsymbol{y}$.
With the rejection sampling formulated in this way, it is natural to consider which other choices for $f_{\boldsymbol{Sc}}$ and $g_{\boldsymbol{Sc}}$ are possible. More general functions $f_{\boldsymbol{Sc}}$ and $g_{\boldsymbol{Sc}}$ can determine both the choice of $\boldsymbol{y} \pm \boldsymbol{Sc}$ and whether or not to reject. In particular, this allows first influencing the probability of constructing $\boldsymbol{z}$ by the rejection condition for $\boldsymbol{z}$. However, note that this requires quite a bit of care as, for the output to follow the desired distribution, changing how often $\boldsymbol{z}$ is constructed must also impact the rejection probability for $\boldsymbol{z}$.

In this work, we consider such more general functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$, with a focus on the case where the input distribution for $\boldsymbol{y}$ is a discrete Gaussian distribution while ensuring that, conditioned on $\boldsymbol{z} \neq \perp$ the emitted $\boldsymbol{z}$ follows the same distribution. By carefully crafting $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$, this allows our scheme to have a significantly lower rejection probability than BLISS.
The rejection failure probability of both our new approach and the approach of BLISS depends on the norm of vectors $\boldsymbol{v}$ used to construct $\boldsymbol{z} = \boldsymbol{y} \pm \boldsymbol{v}$. In particular, it depends on the quotient $\alpha = r/\|\boldsymbol{v}\|$ between the Gaussian parameter r and the norm of the vector $\boldsymbol{v}$ of maximal norm that the rejection sampling can handle. As the rejection sampling handles vectors $\boldsymbol{v} = \boldsymbol{S}\boldsymbol{c}$, the rejection probability depends on the maximal norm of $\boldsymbol{S}\boldsymbol{c}$. The signature size of the scheme essentially scales linearly with $\log(r)$ and thus also with $\log(\alpha)$. As such, if improved rejection sampling allows for a smaller α to be used, it allows for a more compact signature scheme.
When targeting the same rejection probability, our new approach can use a smaller α than previous approaches, and thus allows for smaller signatures. One could hope that this would allow our new approach to use a significantly smaller α, and therefore enable a scheme with significantly smaller signatures. However, decreasing α quickly results in unacceptably large rejection probabilities for our new approach, even though it still does have an advantage over previous rejection sampling methods.
Although our new method has a noticeable advantage for small α, its advantage is much more pronounced for larger α. For instance, with α such that the old approach has a rejection probability of a bit more than 10%, our new approach has to reject less than one in a million signatures. This is a large difference in rejection probability but in practice there is only a small performance difference between rejecting 10% of the time and rejecting only very rarely. However, by additionally considering a new iterative signature construction, we can exploit this significant advantage for larger α to allow for much more compact signatures.

Iterative Rejection Sampling Our new signature scheme is quite similar to BLISS, with an analogous relation between public and private keys, and with both $\boldsymbol{y}$ and $\boldsymbol{z}$ following a discrete Gaussian distribution. Furthermore, the space $\mathcal{C}$ of possible challenges $\boldsymbol{c}$ consist of vectors where all elements are either zero or one and which often are quite sparse. As noted by Ducas [15], and more recently used for the G+G signature scheme [14], this allows signatures from the scheme to be constructed as $\boldsymbol{y} + \boldsymbol{S}\boldsymbol{c}'$ for any $\boldsymbol{c}' \equiv \boldsymbol{c} \mod 2$.
For our work, we make use of this flexibility in how $\boldsymbol{c}'$ can be selected to iteratively rejection sample over the coefficients of $\boldsymbol{c}$. This consists of selecting the sign of each coefficient of $\boldsymbol{c}$ independently, corresponding to either producing $\boldsymbol{y} + \boldsymbol{v}_i$ or $\boldsymbol{y} - \boldsymbol{v}_i$ where $\boldsymbol{v}_i$ is a column of $\boldsymbol{S}$.
To construct $\boldsymbol{z}$, the iterative process is initialized with $\boldsymbol{z}_0 = \boldsymbol{y}$, and a sequence of inputs $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\kappa$ that are columns of $\boldsymbol{S}$ determined by $\boldsymbol{c}$. First, rejection sampling is performed with $\boldsymbol{z}_0$ and the input $\boldsymbol{v}_1$ to get $\boldsymbol{z}_1$ such that, conditioned on $\boldsymbol{z}_1 \neq \perp$, it follows the same distribution as $\boldsymbol{z}_0$. Repeating this leads to a sequence of vectors $\boldsymbol{y} = \boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_\kappa = \boldsymbol{z}$ such

that, for every $i \in \{1, \ldots, \kappa\}$ and conditioned on $\boldsymbol{z}_i \neq \perp$, $\boldsymbol{z}_i$ follows the same distribution as $\boldsymbol{y}$. Thus, the output $\boldsymbol{z}$ follows the desired distribution and there are $a_i \in \{0, 1\}$ such that

$$\boldsymbol{z} = \boldsymbol{y} + \sum_{i=1}^{\kappa} (-1)^{a_i} \boldsymbol{v}_i = \boldsymbol{y} + \boldsymbol{S}\boldsymbol{c}'$$

for some $\boldsymbol{c}' \equiv \boldsymbol{c} \mod 2$.
Each vector $\boldsymbol{v}_i$ added in the iterative steps is a column of $\boldsymbol{S}$ and is therefore expected to be significantly shorter than $\boldsymbol{S}\boldsymbol{c}$. Each rejection sampling step thus handles shorter vectors $\boldsymbol{v}_i$ than a single-step construction of $\boldsymbol{z}$ would have to handle. Although this iterative construction requires multiple steps succeeding, it does allow a larger α to be used for each of the rejection sampling steps. With our new rejection sampling, the larger α more than makes up for multiple steps having to succeed, allowing our scheme to target a significantly narrower output distribution than previous schemes.

1.3 Concrete Signature Scheme

For efficiency and compactness, we use structured lattices as the basis for our signature scheme. In particular, we work with elements in the ring $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$. We additionally limit ourself to n that are powers of two to enable efficient computations through the Number Theoretic Transform (NTT), with a particular focus on $n = 256$.
For our scheme, we let the challenge space $\mathcal{C}$ be a subset of $\mathcal{R}$, selected to only contain ring elements with at most κ non-zero coefficients, with all non-zero elements equal to 1. Furthermore, the secret $\boldsymbol{S}$ of our scheme is a short vector $\boldsymbol{s} \in \mathcal{R}^k$, where k is a relatively small integer typically not larger than 10. Similar to BLISS [16], HAETAE [11, 10] and G+G [14], keys are generated such that $\boldsymbol{A}\boldsymbol{s} = q\boldsymbol{j} \mod 2q$ where $\boldsymbol{A} \in \mathcal{R}^{m \times k}$ is the verification key with $2m \approx k$ and $\boldsymbol{j} = [1, 0, \ldots, 0]^T$ the first unit vector.
A signature for the message μ from the scheme consist of $(\boldsymbol{z}, c)$ where $c \in \mathcal{C}$ and $\boldsymbol{z}$ is a relatively short vector such that

$$\mathcal{H}(\boldsymbol{A}\boldsymbol{z} - qc\boldsymbol{j} \bmod 2q, \mu) = c$$

for $\mathcal{H}$ some cryptographic hash function with output in $\mathcal{C}$. To construct such a signature, the signer samples $\boldsymbol{y}$ from a relatively narrow discrete Gaussian distribution. From this $\boldsymbol{y}$, the signer computes $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{y} \mod 2q$ and lets $c = \mathcal{H}(\boldsymbol{w}, \mu)$.
Next, the vector $\boldsymbol{z}$ is constructed using the previously mentioned iterative rejection sampling. If the rejection sampling fails by producing $\boldsymbol{z} = \perp$, the process is restarted by sampling a new $\boldsymbol{y}$. Otherwise, $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{s}c'$ for some $c' \equiv c \mod 2$ and $\boldsymbol{z}$ follows a relatively narrow discrete Gaussian distribution. Therefore, $\boldsymbol{z}$ is expected to be relatively short and since $\boldsymbol{A}\boldsymbol{s} \equiv q\boldsymbol{j} \mod 2q$

$$\mathcal{H}(\boldsymbol{A}\boldsymbol{z} - qc\boldsymbol{j} \bmod 2q, \mu) = \mathcal{H}(\boldsymbol{w} + qc'\boldsymbol{j} - qc\boldsymbol{j} \bmod 2q, \mu) = \mathcal{H}(\boldsymbol{w}, \mu) = c$$

showing that $(\boldsymbol{z}, c)$ is a valid signature.

Due to our definition of $\mathcal{C}$, $\boldsymbol{sc}$ is equal to the sum of at most κ terms of the form $x^j \boldsymbol{s}$, and the iterative rejection sampling selects the sign for each of these terms individually. Each of these terms have the same length, namely $\|\boldsymbol{s}\|$, and our rejection probability depends on this norm.
For simplicity we ensure that all secret keys of our scheme have the same rejection probability, which is determined by the maximal value of $\|\boldsymbol{s}\|$ over all possible secrets. The secret keys for our scheme are therefore sampled with a length bound B, guaranteeing that $\|\boldsymbol{s}\| \leq B$.
The bound B on the length of the secret key indirectly impacts the size of signatures from the scheme. With the distributions that $\boldsymbol{y}$ and $\boldsymbol{z}$ follow left unchanged, increasing the bound B results in a larger rejection probability. To increase B without altering the rejection probability, $\boldsymbol{y}$ and $\boldsymbol{z}$ must follow wider distributions, leading to larger signatures.

Security The security of our scheme follows from standard arguments in basically the same way as for BLISS.
To argue that it is hard to forge signatures, we assume that the $\boldsymbol{A}$ matrix is constructed from elements that are computationally hard to distinguish from uniformly random. In the random oracle model, it can then be proven that an efficient algorithm for forging signatures implies that the Module Short Integer Solutions (MSIS) problem is easy. Thus, our scheme's security against forgeries can be based upon the assumed hardness of the MSIS problem.
For the MSIS problem, a matrix $\boldsymbol{A}$ is sampled uniformly at random modulo q and the task is to recover a non-zero vector $\boldsymbol{x}$ such that $\boldsymbol{Ax} \equiv \boldsymbol{0} \mod q$ with $\|\boldsymbol{x}\| \leq B$ for some length bound B. The concrete hardness of the MSIS problem is in large part determined by this length bound B. Furthermore, for certain parameter choices, the MSIS problem is at least as hard as standard lattice problems restricted to module lattices [23].
Given an efficient algorithm for forging signatures, the security reduction for the signature scheme can be used to produce short vectors $\boldsymbol{x}$ such that $\boldsymbol{Ax} \equiv \boldsymbol{0} \mod q$. The length of the produced $\boldsymbol{x}$ vector is directly related to the length of the $\boldsymbol{z}$ vector in the forged signatures. If $\boldsymbol{z}$ follows a narrower distribution, the reduction produces shorter vectors and thus solves a harder instance of the MSIS problem.
Besides the width of the distribution of $\boldsymbol{z}$, the hardness of the underlying MSIS problem also depends on the dimension of $\boldsymbol{A} \in \mathcal{R}^{m \times k}$, with a larger m leading to a harder problem. The value of k also has some effect on the hardness of the MSIS problem, with a larger k sometimes making the problem easier. However, a more significant reason to prefer a smaller k is that $\boldsymbol{z}$ is a k-dimensional vector, and the size of signatures therefore directly depend on k.
As mentioned above, we assume that the public key is constructed from elements that are hard to distinguish from uniformly random. There are multiple alternatives for how to sample the secret vector $\boldsymbol{s}$ and the verification key $\boldsymbol{A}$ to achieve this. In particular, there are natural ways to construct such keys based on the assumed hardness of the Module Learning With Errors (MLWE) [23] and NTRU [22] problems. Another

alternative is to create keys based on the assumed hardness of the NTWE problem that was recently introduced by Gärtner [20][3]. There are different benefits for the choice of problem to base the scheme on, and next we describe some of the tradeoffs.

Choice of Structure A natural choice is to base the scheme on the MLWE problem, as done in for instance HAETAE [11, 10] and G+G [14]. In such a scheme, small $\boldsymbol{s}_0 \in \mathcal{R}^{\ell}$ and $\boldsymbol{e} \in \mathcal{R}^{m}$ are sampled as the secret of the scheme and $\boldsymbol{A}_0 \in \mathcal{R}^{m \times \ell}$ is sampled uniformly at random modulo q. The verification key is then given by $\boldsymbol{A}_0, \boldsymbol{b}$ where $\boldsymbol{b} = \boldsymbol{A}_0\boldsymbol{s}_0 + \boldsymbol{e} \mod q$, and the public matrix $\boldsymbol{A}$ and secret vector $\boldsymbol{s}$ are given as

$$\boldsymbol{A} = [q\boldsymbol{j} - 2\boldsymbol{b}, 2\boldsymbol{A}_0, 2\boldsymbol{I}_m] \text{ and } \boldsymbol{s} = [1, \boldsymbol{s}_0^T, \boldsymbol{e}^T]^T.$$

As this results in $\boldsymbol{s} \in \mathcal{R}^{\ell+m+1}$, the $\boldsymbol{z}$ part of signatures is also an $(\ell + m + 1)$-dimensional vector over $\mathcal{R}$. Thus, $\boldsymbol{z}$ can be represented by $(\ell + m + 1)n$ coefficients, all of which are relatively small integers.
For a more compact scheme, we could instead base the scheme on the NTRU problem, similar to for instance BLISS [16] and NTRU+Sign [30]. To target a similar security level, the ring $\mathcal{R}' = \mathbb{Z}[x]/(x^{n'} + 1)$, with $n' = \ell n$ would have to be used. With the original approach of BLISS, the key generation samples small $f, g \in \mathcal{R}'$ as the private key and computes $h = (2g+1)f^{-1} \mod q$ as the public key. The public matrix $\boldsymbol{A}$ and secret vector $\boldsymbol{s}$ are then given by

$$\boldsymbol{A} = [2h, q - 2] \text{ and } \boldsymbol{s} = [f, 2g + 1]^T$$

and, as detailed in [16], this leads to $\boldsymbol{A}\boldsymbol{s} = q \mod 2q$.
For such an NTRU-based scheme, $\boldsymbol{z}$ has $2n' = 2\ell n$ coefficients. Meanwhile, parametrizing an MLWE-based version of the scheme with $m = \ell$ results in $\boldsymbol{z}$ having $(2\ell + 1)n$ coefficients. When targeting similar rejection probability, the coefficients of $\boldsymbol{z}$ for the NTRU-based scheme have similar size as those for an MLWE-based version. As such, signatures of an NTRU-based version of the scheme are somewhat smaller than those of a comparable MLWE-based scheme. However, the NTRU structure limits the flexibility of parameter selection as, in contrast to an MLWE-based scheme, it is not directly possible to select ℓ and m independently from each other. Additionally, for the most efficient NTT calculations n' is limited to powers of two, further limiting the flexibility of parameter selection.
To get the compact signatures of an NTRU-based scheme with the flexibility of an MLWE-based scheme, we instead rely on the NTWE problem. The secret key of such an NTWE-based scheme consists of $\boldsymbol{s}_0 \in \mathcal{R}^{\ell'}$, $\boldsymbol{e} \in \mathcal{R}^{m}$ and $f \in \mathcal{R}$, where $f = 2f_0 + 1$ with the elements of $\boldsymbol{s}_0, \boldsymbol{e}$ and f_0 sampled from some narrow distribution. The verification key of such a

[3] This same problem was first implicitly used for the signature scheme of [2], although without being formally defined. Followup work from this led to a later independent reintroduction of the NTWE problem under the name v-MNTRU [4].

scheme consists of $(\boldsymbol{A}_0, \boldsymbol{b})$ where $\boldsymbol{A}_0 \in \mathcal{R}^{m \times \ell'}$ is sampled with elements uniformly at random modulo q and

$$\boldsymbol{b} = (\boldsymbol{A}_0 \boldsymbol{s}_0 + \boldsymbol{e}) f^{-1} \bmod q.$$

The public matrix and private vector are then given by

$$\boldsymbol{A} = [q\boldsymbol{j} - 2\boldsymbol{b}, 2\boldsymbol{A}_0, 2\boldsymbol{I}_m] \bmod 2q \text{ and } \boldsymbol{s} = [f, \boldsymbol{s}^T, \boldsymbol{e}^T]^T$$

which ensures that $\boldsymbol{A}\boldsymbol{s} = q\boldsymbol{j} \bmod 2q$, as desired.
To target a comparable security level, this NTWE-based version can be parametrized with $\ell' = \ell - 1$ with the coefficients of $\boldsymbol{z}$ still having similar sizes as the ones in the MLWE and NTRU-based versions. Thus, $\boldsymbol{z}$ can be represented by $n(\ell' + m + 1) = n\ell + nm$ relatively small integers and, with $m = \ell$, the size of $\boldsymbol{z}$ is thus essentially the same for this NTWE-based version and for the NTRU-based version. Furthermore, as with the MLWE-based version, we can keep n fixed to some power of two and freely change ℓ' and m in order to alter the targeted security level. As such, this NTWE-based scheme provides the flexibility of an MLWE-based scheme with the compactness of an NTRU-based scheme and it is therefore the focus of this work.

2 Preliminaries

2.1 Notation

We denote matrices by bold upper case letters and vectors by bold lower case letters. We use $\boldsymbol{j} = [1, 0, \ldots, 0]^T$ to denote the first unit vector, with its dimension implicit from the context. The uniform distribution over a set $\mathcal{S}$ we denote by $\mathcal{U}(\mathcal{S})$ and taking a sample x from a distribution $\mathcal{D}$ we denote by $x \leftarrow \mathcal{D}$.
We primarily make use of the Euclidean ℓ_2 norm, which we denote by $\|\cdot\|$. We additionally use the infinity norm and ℓ_1 norm which we denote by $\|\cdot\|_\infty$ and $\|\cdot\|_1$ respectively.
Our signature scheme works in the ring $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$ where n is a power of two. To each element in $\mathcal{R}$ we associate an n-dimensional coefficient vector. Vectors in $\mathcal{R}^k$ can thus be associated with an nk-dimensional vector consisting of the coefficients of the k ring elements. We let the norm $\|\boldsymbol{v}\|$ of a vector $\boldsymbol{v} \in \mathcal{R}^k$ be the norm of this coefficient vector.
We use the standard modular reduction $y = x \bmod p$ such that y is the unique value in $[0, p)$ that satisfies $y = x + kp$ for some integer k. Additionally we use the centered modular reduction $y = x \bmod^{\pm} p$ with y the unique value in $[-p/2, p/2)$ such that $y = x + kp$ for some integer k. We also denote by $\mathrm{LSB}(x)$ the least significant bit of x. All of these functions are naturally extended to vectors, to elements of $\mathcal{R}$ through their coefficient vectors, and to vectors over $\mathcal{R}$.
For a function f and a set $\mathcal{S}$, we let

$$f(\mathcal{S}) = \sum_{x \in \mathcal{S}} f(x).$$

2.2 Signature Schemes

A signature scheme consist of three algorithms $(\text{KeyGen}, \text{Sign}, \text{Verify})$. The KeyGen algorithm outputs a verification key vk and a signing key sk. The signing key sk can be used to sign a message μ via $\mathsf{Sig} \leftarrow \text{Sign}(\mathsf{sk}, \mu)$ and the signature can be verified via $\text{Verify}(\mathsf{Sig}, \mathsf{vk}, \mu)$. With the signature scheme we consider, for valid keys $(\mathsf{vk}, \mathsf{sk}) \leftarrow \text{KeyGen}$, signatures produced via $\mathsf{Sig} \leftarrow \text{Sign}(\mathsf{sk}, \mu)$ always pass verification, as indicated by $\text{Verify}(\mathsf{Sig}, \mathsf{vk}, \mu)$ outputting 1.
For the scheme to be considered secure, it must be hard to forge a signature that passes verification. Three different security notions are relevant for our signature scheme and the proof of its security.
The weakest of the security notions that we consider, which is only used in proving the stronger security properties, is unforgeability under no message attacks (UF-NMA). The advantage of an adversary $\mathcal{A}$ in the UF-NMA security game is given by

$$\Pr\left[\text{Verify}(\mathsf{Sig}, \mathsf{vk}, \mu) = 1 \mid (\mathsf{Sig}, \mu) \leftarrow \mathcal{A}(\mathsf{vk})\right]$$

where vk is a verification key given by KeyGen.
A stronger security notion is unforgeability under chosen message attacks (UF-CMA). The advantage of an adversary in this security game is similar to that of the UF-NMA game, but the adversary is additionally given access to an oracle that signs any message chosen by the adversary. However, the adversary is required to provide a forgery for a message for which the oracle has not provided a signature.
The strongest security notion we consider is strong unforgeability under chosen message attacks (sUF-CMA). The advantage of an adversary in this security game is defined almost the same as for the UF-CMA game, but the adversary is now successful as long as it provides a valid signature which it has not received from the oracle, even if the forgery is for a message which the oracle has provided a different signature for.

2.3 Lattice Assumptions

The security of our scheme can be based on the assumed hardness of the decision NTWE problem and the MSIS problem. The decision NTWE problem and normal form MSIS problem that we rely upon are given by the following definitions.

Definition 1 (NTWE distribution). *Let q be a prime, m and ℓ be positive integers and $\mathcal{D}$ be some distribution over $\mathcal{R}$. Furthermore, let $\boldsymbol{s}$ be a vector in $\mathcal{R}^{\ell}$ and f be an invertible element of $\mathcal{R}_q$. Then, a sample from the NTWE distribution $\mathcal{W}^{m}_{\boldsymbol{s},f,\mathcal{D}}$ is given by $(\boldsymbol{A}, \boldsymbol{b})$ for $\boldsymbol{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{m \times \ell})$ and $\boldsymbol{b} = (\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e})f^{-1} \bmod q$ where $\boldsymbol{e} \leftarrow \mathcal{D}^m$.*

Definition 2 (Decision NTWE problem). *The advantage of an adversary $\mathcal{A}$ against the $NTWE_{q,m,\ell,\mathcal{D}}$ problem is given by*

$$\left| \begin{array}{c} \Pr\left[\mathcal{A}(\boldsymbol{A}, \boldsymbol{b}) = 1 \mid \boldsymbol{A} \leftarrow \mathcal{R}_q^{m \times \ell}, \boldsymbol{b} \leftarrow \mathcal{R}_q^{m}\right] - \\ \Pr\left[\mathcal{A}(\boldsymbol{A}, \boldsymbol{b}) = 1 \mid (\boldsymbol{A}, \boldsymbol{b}) \leftarrow \mathcal{W}^{m}_{\boldsymbol{s},f,\mathcal{D}}\right] \end{array} \right|$$

where $\boldsymbol{s}$ is sampled from $\mathcal{D}^{\ell}$ and $f = 2f_0 + 1$ with f_0 sampled from $\mathcal{D}$ conditioned on f being invertible in $\mathcal{R}_q$.

Definition 3 (Normal form MSIS problem). *Let q be a prime, m and k be positive integers and $B > 0$ be some bound. The advantage of an adversary $\mathcal{A}$ against the $MSIS_{q,m,k,B}$ problem is given by*

$$\Pr\left[0 < \|\boldsymbol{y}\| < B \wedge [\boldsymbol{A}, \boldsymbol{I}_m] \cdot \boldsymbol{y} \equiv 0 \mod q \mid \boldsymbol{A} \leftarrow \mathcal{R}_q^{m \times k}, \boldsymbol{y} \leftarrow \mathcal{A}(\boldsymbol{A})\right].$$

The security of our scheme in the random oracle model can be based on the assumed hardness of the MSIS problem by making use of the forking lemma. However, to construct a more efficient scheme, we instead rely on a special self-target variant of the MSIS problem. This same approach is taken by most other similar lattice-based signature schemes that are constructed using the Fiat-Shamir paradigm.

Such a self-target variant of the MSIS problem could potentially be easier than the ordinary MSIS problem with the same parameters. However, as in previous works, we assume that the self-target MSIS problem actually is as hard as the corresponding MSIS problem. This is motivated by the fact that the only known way to solve this self-target MSIS problem is through solving the corresponding MSIS problem. More specifically, we rely on a bimodal self target MSIS problem defined below, which is the same problem that HAETAE [11, 10] relies on.

This BimodalSelfTargetMSIS problem is parametrized by a size bound $B > 0$ and integers $q, m, \ell > 0$. Furthermore, it depends on a hash function $\mathcal{H}$ from $\mathcal{R}_{2q}^m \times \mathcal{M}$ to $\mathcal{C}$ where $\mathcal{M} \subset \{0,1\}^*$ is a message space and $\mathcal{C} \subseteq \mathcal{R}_2$ is a challenge space.

Definition 4 ($\mathbf{BimodalSelfTargetMSIS}_{\mathcal{H},q,m,\ell,B}$ problem). *An adversary $\mathcal{A}$ against the $BimodalSelfTargetMSIS_{\mathcal{H},q,m,\ell,B}$ problem is given $\boldsymbol{A}_0 \in \mathcal{R}^{m \times \ell}$ and $\boldsymbol{b} \in \mathcal{R}^m$ with elements sampled uniformly at random in $\mathcal{R}_q$. The adversary is to produce $(\boldsymbol{z}, c, \mu) \in (\mathcal{R}^{m+\ell+1}, \mathcal{C}, \mathcal{M})$ such that $\boldsymbol{z} \neq \boldsymbol{0}$, $\|\boldsymbol{z}\| < B$ and*

$$\mathcal{H}(\boldsymbol{A}\boldsymbol{z} - qc\boldsymbol{j} \bmod 2q, \mu) = c$$

where $\boldsymbol{A} = [q\boldsymbol{j} - 2\boldsymbol{b}, 2\boldsymbol{A}_0, 2\boldsymbol{I}] \bmod 2q$.

2.4 Discrete Gaussian Distributions

We use $\rho_r(\boldsymbol{v})$ to denote the Gaussian function $\exp(-\|\boldsymbol{v}\|^2/(2r^2))$, and let $\rho(\boldsymbol{v}) = \rho_1(\boldsymbol{v})$. The discrete Gaussian distribution $\mathcal{D}_{\mathcal{L},r}$ with parameter r over a lattice $\mathcal{L}$ is the distribution where the probability of a vector $\boldsymbol{v} \in \mathcal{L}$ is given by

$$\frac{\rho_r(\boldsymbol{v})}{\sum_{\boldsymbol{w} \in \mathcal{L}} \rho_r(\boldsymbol{w})} = \frac{\rho_r(\boldsymbol{v})}{\rho_r(\mathcal{L})}$$

and the probability of $\boldsymbol{v} \notin \mathcal{L}$ is 0. We also consider discrete Gaussian distributions over the ring $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$, where we sample the coefficient vector of an element in $\mathcal{R}$. This also naturally extends to sampling vectors over $\mathcal{R}$.

3 Our New Rejection Sampling Method

In this section we describe our new rejection sampling method and its advantages over previous methods. Given a sample $\boldsymbol{y}$ from some distribution and some vector $\boldsymbol{v}$ with limited norm, the rejection sampling should either abort and output $\perp$ or output a vector $\boldsymbol{z}$ such that $\boldsymbol{z} = \boldsymbol{y} + k\boldsymbol{v}$ for some odd integer k. Additionally, conditioned on $\boldsymbol{z} \neq \perp$, the output from the rejection sampling should follow some distribution that is independent of $\boldsymbol{v}$. Quite similar to bimodal rejection sampling, unless it aborts, our sampling outputs either $\boldsymbol{y} + \boldsymbol{v}$ or $\boldsymbol{y} - \boldsymbol{v}$.
The novelty of our new method is that the rejection sampling is given more freedom in selecting whether to output $\boldsymbol{y} + \boldsymbol{v}$ or $\boldsymbol{y} - \boldsymbol{v}$. Previous methods for rejection sampling from bimodal distributions first randomly select $\boldsymbol{y} + \boldsymbol{v}$ or $\boldsymbol{y} - \boldsymbol{v}$ with equal probability and then either rejects or outputs the result. We instead select the output via the procedure $R_{\boldsymbol{v}}(\boldsymbol{y})$ in Figure 1 for more general functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$.

$R_{\boldsymbol{v}}(\boldsymbol{y})$

1 : $(a, b) = (f_{\boldsymbol{v}}(\boldsymbol{y}), g_{\boldsymbol{v}}(\boldsymbol{y}))$
2 : $r \leftarrow \mathcal{U}([0, 1))$
3 : **if** $r < a$
4 : **return** $\boldsymbol{y} - \boldsymbol{v}$
5 : **if** $r < a + b$
6 : **return** $\boldsymbol{y} + \boldsymbol{v}$
7 : **return** $\perp$

Fig. 1. New method for rejection sampling, dependent on functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$.

For the output of our new rejection sampling, conditioned on it not being $\perp$, to be independent of the vector $\boldsymbol{v}$, the functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ must satisfy certain properties. The following lemma details conditions on these functions under which the output of $R_{\boldsymbol{v}}(\boldsymbol{y})$ follows some given distribution $\mathcal{Z}$.

Lemma 1. *Let $\mathcal{Y}, \mathcal{Z}$ be some distributions with $p_y(\boldsymbol{y})$ the probability of $\boldsymbol{y} \leftarrow \mathcal{Y}$ and $p_z(\boldsymbol{z})$ the probability of $\boldsymbol{z} \leftarrow \mathcal{Z}$ and let $\boldsymbol{v}$ be some non-zero vector. Furthermore, let $f_{\boldsymbol{v}}(\boldsymbol{z})$ and $g_{\boldsymbol{v}}(\boldsymbol{z})$ be functions and $M \geq 1$ be a parameter such that*

$$f_{\boldsymbol{v}}(\boldsymbol{z}) + g_{\boldsymbol{v}}(\boldsymbol{z}) \leq 1 \tag{1}$$

$$f_{\boldsymbol{v}}(\boldsymbol{z}) \geq 0, \quad g_{\boldsymbol{v}}(\boldsymbol{z}) \geq 0 \tag{2}$$

$$p_y(\boldsymbol{z} + \boldsymbol{v}) f_{\boldsymbol{v}}(\boldsymbol{z} + \boldsymbol{v}) + p_y(\boldsymbol{z} - \boldsymbol{v}) g_{\boldsymbol{v}}(\boldsymbol{z} - \boldsymbol{v}) = \frac{p_z(\boldsymbol{z})}{M} \tag{3}$$

for all $\boldsymbol{z}$. Then, the distribution of $\boldsymbol{z} \leftarrow R_{\boldsymbol{v}}(\boldsymbol{y})$ conditioned on $\boldsymbol{z} \neq \perp$ is $\mathcal{Z}$ and the probability that $\boldsymbol{z} = \perp$ is $1 - 1/M$.

Proof. Based on the properties of $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ we are guaranteed that $R_{\boldsymbol{v}}(\boldsymbol{y})$ returns $\boldsymbol{y} - \boldsymbol{v}$ with probability $f_{\boldsymbol{v}}(\boldsymbol{y})$ and $\boldsymbol{y} + \boldsymbol{v}$ with probability $g_{\boldsymbol{v}}(\boldsymbol{y})$. The probability of an input $\boldsymbol{y}$ is $p_y(\boldsymbol{y})$, and the probability of an output $\boldsymbol{z}$ is therefore

$$p_y(\boldsymbol{z} + \boldsymbol{v})f_{\boldsymbol{v}}(\boldsymbol{z} + \boldsymbol{v}) + p_y(\boldsymbol{z} - \boldsymbol{v})g_{\boldsymbol{v}}(\boldsymbol{z} - \boldsymbol{v}),$$

which via (3) is equal to $p_{\boldsymbol{z}}(\boldsymbol{z})/M$. As this is a scaling of the desired output probability distribution, this gives that conditioned on $\boldsymbol{z} \neq \perp$, the output is distributed as if from $\mathcal{Z}$.
The probability that the procedure does not output $\perp$ is given by the sum of $p_z(\boldsymbol{z})/M$ over all $\boldsymbol{z} \neq \perp$. As p_z is a probability distribution and thus sums to 1, the probability over $\boldsymbol{y}$ that $R_{\boldsymbol{v}}(\boldsymbol{y})$ outputs something other than $\perp$ is $1/M$ and the probability of $\boldsymbol{z} = \perp$ is thus $1 - 1/M$. □

In the following subsections we detail how we construct functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ that fulfill the properties of Lemma 1.
First, in Section 3.1, we consider requirements of such functions when the input and output distributions are the same distribution and when we attempt to target zero rejection probability via setting $M = 1$. In this relatively general setting, we are unable to construct $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ that are guaranteed to fulfill the properties of Lemma 1.
In Section 3.2, we allow for $M > 1$ in a more specialized setting where we adapt the construction of $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ from Section 3.1 to get functions that are guaranteed to fulfill all the properties of Lemma 1. In this more specialized setting, an outcome $\boldsymbol{x}$ of the relevant distribution only depends on $\|\boldsymbol{x}\|$, with the probability of $\boldsymbol{x}$ decreasing as $\|\boldsymbol{x}\|$ increases.
For our signature scheme, we rely on these functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ for the specific case when the relevant distribution is a discrete Gaussian distribution. In Section 3.3, we further analyze the resulting functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ in this setting. In particular, for this setting we somewhat simplify the definition of these functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ and show that they can be efficiently computed to sufficient precision.
Finally, in Section 3.4 we detail the concrete advantage of our new rejection sampling compared to previous rejection sampling methods. In particular, we focus on a comparison to rejection sampling from bimodal Gaussian distributions, as used with BLISS, since this seems like the most relevant comparison.

3.1 Same Input and Output Distributions

For our analysis, we limit ourself to the case where $\mathcal{Z} = \mathcal{Y}$, and thus $p_y(\boldsymbol{x}) = p_z(\boldsymbol{x}) = p(\boldsymbol{x})$ for every $\boldsymbol{x}$. Functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ that satisfy the requirements of Lemma 1 must thus satisfy that

$$\frac{p(\boldsymbol{z})}{M} = p(\boldsymbol{z} + \boldsymbol{v})f_{\boldsymbol{v}}(\boldsymbol{z} + \boldsymbol{v}) + p(\boldsymbol{z} - \boldsymbol{v})g_{\boldsymbol{v}}(\boldsymbol{z} - \boldsymbol{v}). \tag{4}$$

This can be seen as $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ redistributing part of the input probability on the points $\boldsymbol{z} + \boldsymbol{v}$ and $\boldsymbol{z} - \boldsymbol{v}$ to ensure the correct probability for the output $\boldsymbol{z}$.

The functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ also influence the probability of other outputs. In particular, the function $g_{\boldsymbol{v}}$ also distribute part of the probability of the input $\boldsymbol{z}+\boldsymbol{v}$ to ensure that the probability for the output $\boldsymbol{z}+2\boldsymbol{v}$ is correct. Similarly, the function $f_{\boldsymbol{v}}$ distribute part of the probability of the input $\boldsymbol{z}-\boldsymbol{v}$ to ensure the correct probability for the output $\boldsymbol{z}-2\boldsymbol{v}$. The dependence on other inputs cascades, and the functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ thus redistribute the probability of inputs that are an odd multiple of $\boldsymbol{v}$ away from $\boldsymbol{z}$ to ensure the correct probability of outputs that are even multiples of $\boldsymbol{v}$ away from $\boldsymbol{z}$.

For the procedure $R_{\boldsymbol{v}}$ to never reject, the functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ must satisfy the properties of Lemma 1 with $M=1$. For this to be possible for all points related to any given $\boldsymbol{z}$, the probability weight on inputs must equal the probability weight on outputs. This corresponds to the requirement that

$$\sum_{k\in\mathbb{Z}} p(\boldsymbol{z}+2k\boldsymbol{v}) = \sum_{k\in\mathbb{Z}} p(\boldsymbol{z}+(2k+1)\boldsymbol{v})$$

or equivalently that

$$\sum_{k\in\mathbb{Z}} (-1)^k p(\boldsymbol{z}+k\boldsymbol{v}) = 0. \tag{5}$$

However, note that although this requirement is necessary, it is not sufficient for $M=1$ to be possible. For instance, it is impossible to construct valid functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ if $p(\boldsymbol{z}) = t \neq 0$ and $p(\boldsymbol{z}+3\boldsymbol{v}) = t$ while $p(\boldsymbol{z}+k\boldsymbol{v}) = 0$ for all other k.

If (5) holds, there is a natural description of $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ that may satisfy the properties of Lemma 1 with $M=1$. This pair of functions is determined by considering the consequences of (4) being satisfied for $\boldsymbol{z}' = \boldsymbol{z}+2k\boldsymbol{v}$ for every integer $k \neq 0$. With (4) satisfied for every other related output, the probability weight left for $f_{\boldsymbol{v}}(\boldsymbol{z}+\boldsymbol{v})$ and $g_{\boldsymbol{v}}(\boldsymbol{z}-\boldsymbol{v})$ can be determined, and this gives our definition for these functions.

In order for (4) to be satisfied for every output of the form $\boldsymbol{z}+2k\boldsymbol{v}$ with $k>0$, the remaining probability weight on input $\boldsymbol{z}+\boldsymbol{v}$ is

$$p(\boldsymbol{z}+\boldsymbol{v}) - \sum_{k=2}^{\infty} (-1)^k p(\boldsymbol{z}+k\boldsymbol{v}).$$

Letting $f_{\boldsymbol{v}}$ distribute the remaining probability weight to the output $\boldsymbol{z}$ gives that

$$f_{\boldsymbol{v}}(\boldsymbol{z}+\boldsymbol{v}) = \sum_{k=0}^{\infty} (-1)^k \frac{p(\boldsymbol{z}+(k+1)\boldsymbol{v})}{p(\boldsymbol{z}+\boldsymbol{v})}$$

or equivalently

$$f_{\boldsymbol{v}}(\boldsymbol{y}) = \sum_{k=0}^{\infty} (-1)^k \frac{p(\boldsymbol{y}+k\boldsymbol{v})}{p(\boldsymbol{y})}.$$

Similarly, letting the remaining probability weight on the input $\boldsymbol{z}-\boldsymbol{v}$ be distributed by $g_{\boldsymbol{v}}$ to $p(\boldsymbol{z})$ gives that

$$g_{\boldsymbol{v}}(\boldsymbol{y}) = \sum_{k=0}^{\infty} (-1)^k \frac{p(\boldsymbol{y}-k\boldsymbol{v})}{p(\boldsymbol{y})}.$$

These functions fulfill

$$\begin{aligned} f_{\boldsymbol{v}}(\boldsymbol{z}+\boldsymbol{v})p(\boldsymbol{z}+\boldsymbol{v}) + g_{\boldsymbol{v}}(\boldsymbol{z}-\boldsymbol{v})p(\boldsymbol{z}-\boldsymbol{v}) \\ &= \sum_{k=0}^{\infty}(-1)^k(p(\boldsymbol{z}+(k+1)\boldsymbol{v}) + (p(\boldsymbol{z}-(k+1)\boldsymbol{v})) \\ &= p(\boldsymbol{z}) - \sum_{k\in\mathbb{Z}}(-1)^k p(\boldsymbol{z}+k\boldsymbol{v}) = p(\boldsymbol{z}) \end{aligned}$$

with final equality as (5) holds.
For our signature scheme, we can not directly apply this approach as (5) does not hold for every $\boldsymbol{z}$. Furthermore, even with (5) satisfied, these definitions do not guarantee that $f_{\boldsymbol{v}}(\boldsymbol{z}) \geq 0$ and $g_{\boldsymbol{v}}(\boldsymbol{z}) \geq 0$ for every $\boldsymbol{z}$. In the next section, we add some additional requirements on $p(\boldsymbol{z})$ and allow for $M > 1$. This allows us to alter the definitions of $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ so that they do fulfill all the properties of Lemma 1. As we consider $M > 1$, these altered definitions work even if (5) does not hold.

3.2 Norm-Dependent Distribution

In this section, we limit ourself to $p(\boldsymbol{z})$ that depend only on the norm of $\boldsymbol{z}$ and decrease with increasing norm. This is for instance the case for Gaussian distributions, which is the distribution we use for our signature scheme. With such a limitation on $p(\boldsymbol{z})$, we can find a suitable definition for $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ that satisfy all requirements of Lemma 1.
For this analysis, we define the function

$$S_{\boldsymbol{v}}(\boldsymbol{y}) = \sum_{k=0}^{\infty}(-1)^k \frac{p(\boldsymbol{y}+k\boldsymbol{v})}{p(\boldsymbol{y})}$$

which is the definition of $f_{\boldsymbol{v}}(\boldsymbol{y})$ from the previous section. Furthermore, as $p(\boldsymbol{y})$ is only determined by the norm of $\boldsymbol{y}$, we have that

$$S_{\boldsymbol{v}}(-\boldsymbol{y}) = \sum_{k=0}^{\infty}(-1)^k \frac{p(-\boldsymbol{y}+k\boldsymbol{v})}{p(-\boldsymbol{y})} = \sum_{k=0}^{\infty}(-1)^k \frac{p(\boldsymbol{y}-k\boldsymbol{v})}{p(\boldsymbol{y})}$$

which is the definition of $g_{\boldsymbol{v}}(\boldsymbol{y})$ from the previous section.
We are guaranteed that either

$$\|\boldsymbol{y}+(k+1)\boldsymbol{v}\|^2 \geq \|\boldsymbol{y}+k\boldsymbol{v}\|^2 \text{ or } \|\boldsymbol{y}-(k+1)\boldsymbol{v}\|^2 \geq \|\boldsymbol{y}-k\boldsymbol{v}\|^2$$

for every $k \geq 0$ and we know that $p(\boldsymbol{x})$ is decreasing with increasing $\|\boldsymbol{x}\|$. From this it follows that either $S_{\boldsymbol{v}}(\boldsymbol{y}) \geq 0$ or $S_{\boldsymbol{v}}(-\boldsymbol{y}) \geq 0$, since for at least one of these functions, the absolute values of terms in the sum are strictly decreasing. The idea is to define both $f_{\boldsymbol{v}}(\boldsymbol{y})$ and $g_{\boldsymbol{v}}(\boldsymbol{y})$ based on the one of $S_{\boldsymbol{v}}(\boldsymbol{y})$ and $S_{\boldsymbol{v}}(-\boldsymbol{y})$ that is guaranteed to be positive. This corresponds to deciding whether to sum up positive or negative multiples of $\boldsymbol{v}$ based on which choice moves the sum away from $\mathbf{0}$.
In particular, if $\langle \boldsymbol{y}, \boldsymbol{v} \rangle \geq \|\boldsymbol{v}\|^2$, then we let $f_{\boldsymbol{v}}(\boldsymbol{y}) = S_{\boldsymbol{v}}(\boldsymbol{y})/M \geq 0$ for some $M \geq 1$. In order for (4) to be fulfilled for $\boldsymbol{z} = \boldsymbol{y} - \boldsymbol{v}$, we then require that

$$g_{\boldsymbol{v}}(\boldsymbol{y}-2\boldsymbol{v}) = \frac{p(\boldsymbol{y}-\boldsymbol{v}) - S_{\boldsymbol{v}}(\boldsymbol{y})p(\boldsymbol{y})}{Mp(\boldsymbol{y}-2\boldsymbol{v})} = \frac{1 - S_{\boldsymbol{v}}(\boldsymbol{y}-2\boldsymbol{v})}{M}.$$

Similarly, if $\langle \boldsymbol{y}, \boldsymbol{v} \rangle < \|\boldsymbol{v}\|^2$, we instead let $f_{\boldsymbol{v}}(\boldsymbol{y}) = (1 - S_{\boldsymbol{v}}(-\boldsymbol{y}))/M$ which leads to the requirement that

$$g_{\boldsymbol{v}}(\boldsymbol{y} - 2\boldsymbol{v}) = \frac{p(-\boldsymbol{y} + \boldsymbol{v}) + (S_{\boldsymbol{v}}(-\boldsymbol{y}) - 1)p(\boldsymbol{y})}{Mp(-\boldsymbol{y} + 2\boldsymbol{v})} = \frac{S_{\boldsymbol{v}}(-\boldsymbol{y} + 2\boldsymbol{v})}{M}.$$

in order for (4) to be fulfilled for $\boldsymbol{z} = \boldsymbol{y} - \boldsymbol{v}$.
These definitions for $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ are detailed in the theorem below, where we prove that, for a suitable M, these functions fulfill all requirements of Lemma 1. Note that if (5) is fulfilled for all relevant $\boldsymbol{y}$, then these functions satisfy the requirements of Lemma 1 with $M = 1$ and we recover the same functions as considered in the previous section.

Theorem 1. *Let $\mathcal{K}_{\boldsymbol{v}}$ be the set of $\boldsymbol{y}$ such that $|\langle \boldsymbol{y}, \boldsymbol{v} \rangle| \leq \|\boldsymbol{v}\|^2$ and let $M \geq 1$ be such that*

$$M \geq \max_{\boldsymbol{y} \in \mathcal{K}_{\boldsymbol{v}}} 1 - \sum_{k \in \mathbb{Z}} (-1)^k \frac{p(\boldsymbol{y} + k\boldsymbol{v})}{p(\boldsymbol{y})}.$$

Then, the functions

$$f_{\boldsymbol{v}}(\boldsymbol{y}) = \begin{cases} \dfrac{S_{\boldsymbol{v}}(\boldsymbol{y})}{M} & \text{If } \langle \boldsymbol{y}, \boldsymbol{v} \rangle \geq \|\boldsymbol{v}\|^2 \\ \dfrac{1 - S_{\boldsymbol{v}}(-\boldsymbol{y})}{M} & \text{If } \langle \boldsymbol{y}, \boldsymbol{v} \rangle < \|\boldsymbol{v}\|^2 \end{cases}$$

and

$$g_{\boldsymbol{v}}(\boldsymbol{y}) = \begin{cases} \dfrac{1 - S_{\boldsymbol{v}}(\boldsymbol{y})}{M} & \text{If } \langle \boldsymbol{y}, \boldsymbol{v} \rangle \geq -\|\boldsymbol{v}\|^2 \\ \dfrac{S_{\boldsymbol{v}}(-\boldsymbol{y})}{M} & \text{If } \langle \boldsymbol{y}, \boldsymbol{v} \rangle < -\|\boldsymbol{v}\|^2 \end{cases}$$

fulfill all the requirements of Lemma 1.

Proof. We begin by showing that (3) holds for these definitions of $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$. For $\boldsymbol{z}$ such that $\langle \boldsymbol{z}, \boldsymbol{v} \rangle \geq 0$, we have $\langle \boldsymbol{z} - \boldsymbol{v}, \boldsymbol{v} \rangle \geq -\left\|\boldsymbol{v}^2\right\|$ and $\langle \boldsymbol{z} + \boldsymbol{v}, \boldsymbol{v} \rangle \geq \left\|\boldsymbol{v}^2\right\|$. As such, we have

$$\begin{aligned} &p(\boldsymbol{z} + \boldsymbol{v}) f_{\boldsymbol{v}}(\boldsymbol{z} + \boldsymbol{v}) + p(\boldsymbol{z} - \boldsymbol{v}) g_{\boldsymbol{v}}(\boldsymbol{z} - \boldsymbol{v}) \\ &\quad = \frac{S_{\boldsymbol{v}}(\boldsymbol{z} + \boldsymbol{v}) p(\boldsymbol{z} + \boldsymbol{v}) + (1 - S_{\boldsymbol{v}}(\boldsymbol{z} - \boldsymbol{v})) p(\boldsymbol{z} - \boldsymbol{v})}{M} \\ &\quad = \frac{p(\boldsymbol{z} - \boldsymbol{v})}{M} + \sum_{k=0}^{\infty} (-1)^k \frac{p(\boldsymbol{z} + (k+1)\boldsymbol{v}) - p(\boldsymbol{z} + (k-1)\boldsymbol{v})}{M} \\ &\quad = \frac{p(\boldsymbol{z})}{M} \end{aligned}$$

as expected. Furthermore, when $\langle \boldsymbol{z}, \boldsymbol{v} \rangle < 0$, we have $\langle \boldsymbol{z} - \boldsymbol{v}, \boldsymbol{v} \rangle < -\left\|\boldsymbol{v}^2\right\|$ and $\langle \boldsymbol{z} + \boldsymbol{v}, \boldsymbol{v} \rangle < \left\|\boldsymbol{v}^2\right\|$ and thus

$$\begin{aligned} &p(\boldsymbol{z} + \boldsymbol{v}) f_{\boldsymbol{v}}(\boldsymbol{z} + \boldsymbol{v}) + p(\boldsymbol{z} - \boldsymbol{v}) g_{\boldsymbol{v}}(\boldsymbol{z} - \boldsymbol{v}) \\ &\quad = \frac{(1 - S_{\boldsymbol{v}}(-\boldsymbol{z} - \boldsymbol{v})) p(\boldsymbol{z} + \boldsymbol{v}) + S_{\boldsymbol{v}}(-\boldsymbol{z} + \boldsymbol{v}) p(\boldsymbol{z} - \boldsymbol{v})}{M} \\ &\quad = \frac{p(\boldsymbol{z} + \boldsymbol{v})}{M} + \sum_{k=0}^{\infty} (-1)^k \frac{p(\boldsymbol{z} - (k-1)\boldsymbol{v}) - p(\boldsymbol{z} - (k+1)\boldsymbol{v})}{M} \\ &\quad = \frac{p(\boldsymbol{z})}{M} \end{aligned}$$

as expected.
Next, we show that $f_{\boldsymbol{v}}(\boldsymbol{y}) \geq 0$ and $g_{\boldsymbol{v}}(\boldsymbol{y}) \geq 0$ for every $\boldsymbol{y}$. This follows from showing that, when $\langle \boldsymbol{y}, \boldsymbol{v} \rangle \geq -\|\boldsymbol{v}\|^2$ then $S_{\boldsymbol{v}}(\boldsymbol{y}) \leq 1$ and when $\langle \boldsymbol{y}, \boldsymbol{v} \rangle \geq \|\boldsymbol{v}\|^2$ then $S_{\boldsymbol{v}}(\boldsymbol{y}) \geq 0$. To this end, we first note that

$$\|\boldsymbol{y} + k\boldsymbol{v}\|^2 = \|\boldsymbol{y}\|^2 + 2k\langle \boldsymbol{y}, \boldsymbol{v} \rangle + k^2\|\boldsymbol{v}\|^2$$

and see that if $\langle \boldsymbol{y}, \boldsymbol{v} \rangle \geq 0$, then $\|\boldsymbol{y} + k\boldsymbol{v}\|$ is greater than $\|\boldsymbol{y} + (k-1)\boldsymbol{v}\|$ for every $k \geq 1$.
When $\langle \boldsymbol{y}, \boldsymbol{v} \rangle \geq 0$, and thus also when $\langle \boldsymbol{y}, \boldsymbol{v} \rangle \geq \|\boldsymbol{v}\|^2$, we have

$$S_{\boldsymbol{v}}(\boldsymbol{y}) = \sum_{k=0}^{\infty} (-1)^k \frac{p(\boldsymbol{y} + k\boldsymbol{v})}{p(\boldsymbol{y})} = \sum_{k=0}^{\infty} \frac{p(\boldsymbol{y} + 2k\boldsymbol{v}) - p(\boldsymbol{y} + (2k+1)\boldsymbol{v})}{p(\boldsymbol{y})} \geq 0$$

as every term in the second sum is non-negative. Furthermore, when $\langle \boldsymbol{y}, \boldsymbol{v} \rangle \geq -\|\boldsymbol{v}\|^2$, we have

$$\begin{aligned} S_{\boldsymbol{v}}(\boldsymbol{y}) &= \sum_{k=0}^{\infty} (-1)^k \frac{p(\boldsymbol{y} + k\boldsymbol{v})}{p(\boldsymbol{y})} = 1 - \sum_{k=1}^{\infty} (-1)^{k+1} \frac{p(\boldsymbol{y} + k\boldsymbol{v})}{p(\boldsymbol{y})} \\ &= 1 - S_{\boldsymbol{v}}(\boldsymbol{y} + \boldsymbol{v}) \cdot \frac{p(\boldsymbol{y} + \boldsymbol{v})}{p(\boldsymbol{y})} \end{aligned}$$

and this is upper bounded by 1 as $S_{\boldsymbol{v}}(\boldsymbol{y} + \boldsymbol{v}) \geq 0$ since $\langle \boldsymbol{y} + \boldsymbol{v}, \boldsymbol{v} \rangle \geq 0$.
Finally, left to show is only that $f_{\boldsymbol{v}}(\boldsymbol{y}) + g_{\boldsymbol{v}}(\boldsymbol{y}) \leq 1$ for every $\boldsymbol{y}$. To this end, we first note that when $|\langle \boldsymbol{y}, \boldsymbol{v} \rangle| > \|\boldsymbol{v}\|^2$ this is trivially true as

$$f_{\boldsymbol{v}}(\boldsymbol{y}) + g_{\boldsymbol{v}}(\boldsymbol{y}) = \begin{cases} \dfrac{S_v(\boldsymbol{y}) + 1 - S_{\boldsymbol{v}}(\boldsymbol{y})}{M} & \text{If } \langle \boldsymbol{y}, \boldsymbol{v} \rangle > \|\boldsymbol{v}\|^2 \\ \dfrac{S_v(-\boldsymbol{y}) + 1 - S_{\boldsymbol{v}}(-\boldsymbol{y})}{M} & \text{If } \langle \boldsymbol{y}, \boldsymbol{v} \rangle < -\|\boldsymbol{v}\|^2 \end{cases} = \frac{1}{M}.$$

Meanwhile, when $|\langle \boldsymbol{y}, \boldsymbol{v} \rangle| \leq \|\boldsymbol{v}\|^2$, we have that

$$\begin{aligned} f_{\boldsymbol{v}}(\boldsymbol{y}) + g_{\boldsymbol{v}}(\boldsymbol{y}) &= \frac{2 - S_{\boldsymbol{v}}(\boldsymbol{y}) - S_{\boldsymbol{v}}(-\boldsymbol{y})}{M} \\ &= \frac{1}{M}\left(1 - \sum_{k \in \mathbb{Z}} (-1)^k \frac{p(\boldsymbol{y} + k\boldsymbol{v})}{p(\boldsymbol{y})}\right) \leq 1. \end{aligned}$$

with final inequality due to the definition of M in the theorem statement. □

3.3 Discrete Gaussian Distributions

To actually make use of the definitions of $f_{\boldsymbol{v}}(\boldsymbol{y})$ and $g_{\boldsymbol{v}}(\boldsymbol{y})$ in Theorem 1 we must be able to efficiently compute $S_{\boldsymbol{v}}(\boldsymbol{y})$ with sufficiently high precision. In this section, we show that this is possible when we work with a discrete Gaussian distribution, which is the case we focus on in this work.
When considering a discrete Gaussian distribution, $p(\boldsymbol{y})$ is proportional to $\rho_r(\boldsymbol{y})$ for some Gaussian parameter r. As such, we have that

$$S_{\boldsymbol{v}}(\boldsymbol{y}) = \sum_{k \geq 0} (-1)^k \frac{\rho_r(\boldsymbol{y} + k\boldsymbol{v})}{\rho_r(\boldsymbol{y})}$$

and we want to efficiently compute a sufficiently good approximation of this $S_{\boldsymbol{v}}(\boldsymbol{y})$. The following lemma bounds the error incurred by only including the t first terms of the infinite sum in the expression for $S_{\boldsymbol{v}}(\boldsymbol{y})$, in the regime where its value is relevant to us. This shows that only accounting for a few terms of this sum provides a sufficiently good approximation of $S_{\boldsymbol{v}}(\boldsymbol{y})$. In particular, only including the first 20 terms of the sum results in a value that differs significantly less than 2^{-256} from the true value of the infinite sum, even for the worst case[4] where $\|\boldsymbol{v}\| = 1$. Since the lemma follows from simple calculations, we omit the proof from here and instead include it in Appendix D.

Lemma 2. *The function $S_{\boldsymbol{v}}(\boldsymbol{y})$ only depends on $\langle \boldsymbol{y}, \boldsymbol{v} \rangle$ and $\|\boldsymbol{v}\|^2$. Furthermore, for arbitrary $t \geq 1$, fixed $\boldsymbol{v}$ and with $\langle \boldsymbol{y}, \boldsymbol{v} \rangle \geq -\|\boldsymbol{v}\|^2$ then*

$$\left| S_{\boldsymbol{v}}(\boldsymbol{y}) - \sum_{k=0}^{t} (-1)^k \frac{\rho_r(\boldsymbol{y} + k\boldsymbol{v})}{\rho_r(\boldsymbol{y})} \right| \leq \frac{\rho_r(t\boldsymbol{v})}{\rho_r(\boldsymbol{v}) - \rho_r(2\boldsymbol{v})}$$

To actually compute $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$, we must also have an efficient way to compute a value for M. The value of M must be no smaller than the infinite sum in Theorem 1 and we determine a simple expression that satisfy this in Lemma 3 below. This bound is very tight for relevant parameters, and there is therefore no significant downside to using this expression for M. However, note that the main purpose of this is to simplify the expression for M. We therefore do not include the proof here, instead including it in Appendix D where it is proven using the Poisson summation formula.

Lemma 3. *Let $\mathcal{K}_{\boldsymbol{v}}$ be the set of $\boldsymbol{y}$ such that $|\langle \boldsymbol{y}, \boldsymbol{v} \rangle| \leq \|\boldsymbol{v}\|^2$ and let $\alpha = r/\|\boldsymbol{v}\|$. Then*

$$\max_{\boldsymbol{y} \in \mathcal{K}_{\boldsymbol{v}}} 1 - \sum_{k \in \mathbb{Z}} (-1)^k \frac{\rho_r(\boldsymbol{y} + k\boldsymbol{v})}{\rho_r(\boldsymbol{y})} \leq 1 + \frac{2\alpha\sqrt{2\pi}\rho(\pi\alpha)}{\rho_\alpha(1) \cdot (1 - \rho_1(2\pi\alpha))} = M_\alpha$$

and M_α is strictly decreasing with α.

This finally gives the expression for the functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ that we use for the rejection sampling via $R_{\boldsymbol{v}}$ for our signature scheme. These definitions for $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ are given in Corollary 1 below, which follows from combining Theorem 1 with Lemma 3.

Corollary 1. *Let $\alpha > 0$ and let*

$$M = M_\alpha = 1 + \frac{2\alpha\sqrt{2\pi}\rho(\pi\alpha)}{\rho_\alpha(1) \cdot (1 - \rho(2\pi\alpha))}.$$

For every vector $\boldsymbol{v}$ such that $\alpha \leq r/\|\boldsymbol{v}\|$, the functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ defined via

$$f_{\boldsymbol{v}}(\boldsymbol{y}) = \begin{cases} \dfrac{S_{\boldsymbol{v}}(\boldsymbol{y})}{M} & \text{If } \langle \boldsymbol{y}, \boldsymbol{v} \rangle \geq \|\boldsymbol{v}\|^2 \\ \dfrac{1 - S_{\boldsymbol{v}}(-\boldsymbol{y})}{M} & \text{If } \langle \boldsymbol{y}, \boldsymbol{v} \rangle < \|\boldsymbol{v}\|^2 \end{cases}$$

[4] The special case $\boldsymbol{v} = \boldsymbol{0}$ is never relevant for our rejection sampling.

and

$$g_{\boldsymbol{v}}(\boldsymbol{y}) = \begin{cases} \dfrac{1 - S_{\boldsymbol{v}}(\boldsymbol{y})}{M} & \text{If } \langle \boldsymbol{y}, \boldsymbol{v} \rangle \geq -\|\boldsymbol{v}\|^2 \\ \dfrac{S_{\boldsymbol{v}}(-\boldsymbol{y})}{M} & \text{If } \langle \boldsymbol{y}, \boldsymbol{v} \rangle < -\|\boldsymbol{v}\|^2 \end{cases}$$

fulfill all the requirements of Lemma 1.

3.4 Benefit of Our Method

Using the functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$, as given by Corollary 1 to rejection sample with $R_{\boldsymbol{v}}$ leads to a repetition rate of M. We want to compare this M with the repetition rate of previous works for lattice-based signatures constructed using the Fiat-Shamir with aborts methodology. In particular, we compare our repetition rate to that of schemes relying on rejection sampling from bimodal Gaussian distributions.

For this comparison, we define the set $\mathcal{V}$ to contain all possible vectors $\boldsymbol{v}$ that we want the functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ given by Corollary 1 to handle. The rejection rate of both our approach and previous ones is dependent on $\alpha = r/\|\boldsymbol{w}\|$, where $\boldsymbol{w}$ is the element of $\mathcal{V}$ of maximal norm. In this setting, bimodal Gaussian schemes such as BLISS [16], have a repetition rate of $\exp(1/(2\alpha^2)) = 1/\rho_\alpha(1)$, and the expected number of rejections is thus $1/\rho_\alpha(1) - 1$.

In order to illustrate the benefit of our new method, we provide a plot in Figure 2 with the logarithm of the expected number of rejections for our new method compared to the same value for BLISS. For the parametrizations in BLISS, a value of α between 0.5 and 1 is used. As can be seen from the plot in Figure 2, our new method has a noticeable advantage in the rejection rate for the values of α used for BLISS. From this it also follows that, with the same rejection rate, our new method allows for a parametrization with somewhat smaller α, leading to somewhat smaller signatures.

Note that the advantage in terms of rejection rate of our method increase as α increase. For relatively large α the ordinary bimodal Gaussian rejection sampling method still has to reject relatively frequently, whereas the rejection rate for our new method quickly approaches 0 with larger α. By itself, this does not provide a significant advantage as, from a performance perspective, there is not that big of a difference between rejecting one out of ten signatures compared to rejecting signatures only very rarely.

The low rejection rate for larger α can, however, be used to ensure that even if the sampling is repeated multiple times, there is still a high probability that all repetitions succeed. This allows our new method to gain a significant advantage over previous methods, as it enables an iterative construction of $\boldsymbol{z}$. As described in Section 1.2, this is accomplished by iteratively performing rejection sampling over each of the coefficients of $\boldsymbol{c}$. This iterative process is initialised with $\boldsymbol{z}_0 = \boldsymbol{y}$ and $k \leq \kappa$ vectors $\boldsymbol{c}_i$ such that $\boldsymbol{c} = \sum_i \boldsymbol{c}_i$ while each $\boldsymbol{c}_i$ consists of zeroes and a single 1. The iterative process then constructs $\boldsymbol{z}_i = \boldsymbol{z}_{i-1} \pm \boldsymbol{S}\boldsymbol{c}_i$ for each $i \in 1, \ldots, k$, with the final output of the rejection sampling given by $\boldsymbol{z} = \boldsymbol{z}_k$.

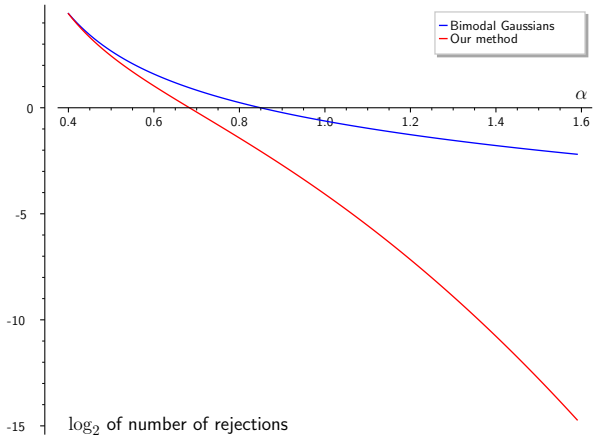

Fig. 2. Base two logarithm of the expected number of rejections for each signature with normal bimodal Gaussian rejection sampling and with our new method.

Similar to without the iterative approach, the functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ used for the iterative rejection sampling are defined via Corollary 1. However, compared to the non-iterative approach, the set $\mathcal{V}$ that the functions has to handle vectors $\boldsymbol{v}$ from is different. With the iterative rejection sampling $\mathcal{V}$ consists of all possible columns of $\boldsymbol{S}$, whereas without it, the set consists of all possible values of $\boldsymbol{Sc}$. As the longest column of $\boldsymbol{S}$ is significantly shorter than the longest vector of the form $\boldsymbol{Sc}$, this allows the functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ to be defined with $M = M_\alpha$ for a significantly larger α.

Another benefit of our new method for rejection sampling is that the rejection probability decrease exponentially with increasing α. As such, it is easy to select parameters that target a negligible rejection probability and therefore allow the rejection condition to be safely ignored. To quantify this, we note that the rejection probability is given by

$$\begin{aligned} 1 - \frac{1}{M} &= \frac{2\alpha\sqrt{2\pi}\rho(\pi\alpha)}{\rho_\alpha(1) \cdot (1 - \rho(2\pi\alpha)) + 2\alpha\sqrt{2\pi}\rho(\pi\alpha)} \\ &\leq \frac{2\alpha\sqrt{2\pi}\rho(\pi\alpha)}{\rho_\alpha(1) \cdot (1 - \rho(2\pi\alpha))} < 4\alpha\sqrt{2\pi}\rho(\pi\alpha) \end{aligned}$$

with the final inequality holding when $\alpha \geq 1$. From this, it follows that

$$\log\left(1 - \frac{1}{M}\right) = \log(4\alpha\sqrt{2\pi}) - \log(e)\pi^2\alpha^2/2 \tag{6}$$

from which we can see that $\alpha \geq 4$ leads to a rejection probability that is smaller than 2^{-108}.

A signature scheme parametrized with $\alpha \geq 4$ could thus quite likely be used to sign on the order of 2^{100} messages without ever triggering the rejection condition. Furthermore, only a little information about the secret

is leaked if a single signature that should have been rejected is emitted by the signature scheme. As such, a scheme parametrized with $\alpha \geq 4$ can be implemented while ignoring the rejection condition, without this significantly impacting the security of the scheme.

4 Signature Scheme

The high-level construction of our signature scheme is the same as that of BLISS. However, our scheme differs from BLISS by using our new method for rejection sampling in an iterative manner to construct the signatures. Additionally, the public key of our scheme is constructed based on the NTWE problem whereas BLISS is based on the NTRU problem.
Due to the iterative construction of our signatures, there are several separate chances for rejection, once for each of the at most κ non-zero coefficient of c. However, the iterative construction also allows for a larger α to be used and therefore results in a lower rejection probability. Although having to repeat the sampling κ times increases the total rejection probability, when using our new rejection sampling, this is more than compensated for by the iterative construction enabling a larger α to be used. As such, when targeting a similar total rejection probability, the iterative construction allows our scheme to be parametrized with significantly smaller parameters, resulting in a more compact scheme.
Note that our iterative construction of signatures could also be used with previous rejection sampling methods to enable a larger α. However, for previous methods, the larger α typically does not compensate for the increase to total rejection probability due to having to repeat the sampling κ times. Thus, although a similar iterative construction of signatures is possible also with previous rejection sampling methods, it would not be beneficial.
Our signature scheme makes use of the functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ from Corollary 1 to implement the rejection sampling procedure $R_{\boldsymbol{v}}(\boldsymbol{y})$ described in Figure 1. For the definition of these functions, we let $\mathcal{V}$ be the set of all possible secret vectors $\boldsymbol{s}$, ensuring that the scheme has the same rejection probability for all possible secrets. We furthermore ensure that only keys with $\|\boldsymbol{s}\| \leq B_{\mathrm{k}}$ are generated, and thus $\alpha = r/B_{\mathrm{k}}$ is used to determine the value of $M = M_\alpha$.
To illustrate the high level structure of the scheme, we present an uncompressed version of our scheme in Figure 3 and briefly describe it in the next section. As described in Section 4.2, significantly more compact signatures are possible through utilizing compression techniques similar to those utilized in many previous works [3, 10, 14, 26]. Although this compression somewhat complicates the description of the scheme, it is relatively standard and is very similar to that of previous schemes. Because of this, most details of the compression are left to Appendix A.

4.1 Uncompressed Signature Scheme

For the key-generation $\boldsymbol{A}_0$ and $\boldsymbol{b}$ are generated as an NTWE instance, and, assuming the hardness of the decision NTWE problem, the pair

$(\boldsymbol{A}_0, \boldsymbol{b})$ are computationally indistinguishable from uniformly random. The verification key $\boldsymbol{A}$ is constructed from $\boldsymbol{A}_0$ and $\boldsymbol{b}$ whereas the secret key $\boldsymbol{s}$ is constructed from the secret of the NTWE instance, with these keys constructed to fulfill that $\boldsymbol{As} \equiv q\boldsymbol{j} \mod 2q$.
For signing a message, a relatively short vector $\boldsymbol{y}$ is sampled and a commitment $\boldsymbol{w} = \boldsymbol{Ay} \mod 2q$ is computed. From this, and the message to be signed, a challenge c is derived using a random oracle $\mathcal{H} : \mathcal{R}_q^m \times \mathcal{M} \to \mathcal{C}$ where $\mathcal{M}$ is the message space and $\mathcal{C}$ is some challenge space. In practice this random oracle is instantiated from a secure hash function combined with a way to sample from the challenge space, in the same way as for HAETAE.
A signature consist of this challenge, and a short vector $\boldsymbol{z}$ such that

$$\boldsymbol{Az} \equiv \boldsymbol{w} + qc\boldsymbol{j} \mod 2q. \tag{7}$$

To construct $\boldsymbol{z}$, the rejection sampling procedure $R_{\boldsymbol{v}}$ in Figure 1 is used with the functions $f_{\boldsymbol{v}}$ and $g_{\boldsymbol{v}}$ from Corollary 1. The vectors $\boldsymbol{v}$ that are used are given by $\boldsymbol{s}x^j$, for each monomial x^j in c. The end result is $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{s}c'$ for some ternary $c' \equiv c \mod 2$ which ensures that (7) holds. Verification verifies that $\boldsymbol{z}$ is sufficiently short, and implicitly verifies that (7) holds by checking that

$$\mathcal{H}(\boldsymbol{Az} - qc\boldsymbol{j} \bmod 2q, \mu) = \mathcal{H}(\boldsymbol{w}, \mu) = c.$$

Challenge Space The challenge space $\mathcal{C}$ of our signature scheme contains elements in $\mathcal{R}$ with at most κ non-zero coefficients, and with $\mathcal{C}$ that contains a sufficiently large number of possible challenges. For most parametrizations, $\mathcal{C}$ consist of elements with exactly κ coefficients equal to one and the remaining coefficients equal to zero.
For the parametrizations which target the highest security level, the challenge space $\mathcal{C}$ is selected somewhat differently in order to increase its size. As with HAETAE [11, 10], for these parametrizations $\mathcal{C}$ instead consist of exactly half of all possible degree $n-1$ polynomials with only 0 and 1 coefficients. Furthermore, the challenge polynomials all contain at most $n/2$ non-zero coefficients. In this case, the challenge space thus contains 2^{n-1} different elements. For details on exactly which polynomials are included, and how to sample from this challenge space, we refer to Section 3.3 of [11].
In order to ensure that the rejection sampling does not influence the distribution of challenges c in signatures that are output, we ensure that every challenge c has the same rejection probability. Over a random $\boldsymbol{y}$, the function $R_{\boldsymbol{v}}(\boldsymbol{y})$ has the same rejection rate for every $\boldsymbol{v}$, and thus the rejection rate for a challenge c is influenced only by the number of non-zero coefficients in c.
As mentioned above, for the highest security level, the number of non-zero coefficients vary and challenges with fewer non-zero coefficients require fewer iterative steps for the rejection sampling. As each iterative step has the same rejection rate, if not handled separately, challenges with fewer non-zero coefficients would be more likely in output signatures. To ensure that every challenge c is equally likely, signatures that

KeyGen

1 : $A_0 \leftarrow \mathcal{U}(\mathcal{R}_q)^{m \times \ell}$
2 : $\boldsymbol{s}_0 \leftarrow \mathcal{D}^{\ell}_{\mathcal{R},\sigma}$
3 : $\boldsymbol{e} \leftarrow \mathcal{D}^{m}_{\mathcal{R},\sigma}$
4 : $f_0 \leftarrow \mathcal{D}_{\mathcal{R},\sigma}$
5 : $f = 2f_0 + 1$
6 : **if** f is not invertible in $\mathcal{R}_q$
7 : Goto line 4
8 : $\boldsymbol{b} = (\boldsymbol{A}_0\boldsymbol{s}_0 + \boldsymbol{e})/f \bmod q$
9 : $\boldsymbol{A} = [q\boldsymbol{j} - 2\boldsymbol{b}, 2\boldsymbol{A}_0, 2\boldsymbol{I}_m]$
10 : $\boldsymbol{s} = [f, \boldsymbol{s}_0, \boldsymbol{e}]$
11 : **if** $\|\boldsymbol{s}\| \geq B_{\mathrm{k}}$
12 : Goto line 2
13 : **return** $vk = \boldsymbol{A}, sk = \boldsymbol{s}$

Verify$((\boldsymbol{z}, c), \boldsymbol{A}, \mu)$

1 : Reject if $\|\boldsymbol{z}\| > B_{\mathrm{s}}$
2 : Accept if $\mathcal{H}(\boldsymbol{A}\boldsymbol{z} - qc\boldsymbol{j} \bmod 2q, \mu) = c$

Sign$(\boldsymbol{A}, \boldsymbol{s}, \mu)$

1 : $\boldsymbol{y} \leftarrow \mathcal{D}^{\ell+m+1}_{\mathcal{R},r}$
2 : $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{y} \bmod 2q$
3 : $c = \mathcal{H}(\boldsymbol{w}, \mu)$
4 : $\boldsymbol{z} = \mathrm{RejectSample}(\boldsymbol{y}, c)$
5 : **if** $\boldsymbol{z} = \perp$ or $\|\boldsymbol{z}\| > B_{\mathrm{s}}$
6 : Goto line 1
7 : **return** $(\boldsymbol{z}, c)$

RejectSample$(\boldsymbol{z}_0, c)$

1 : $k = 0$
2 : **for** Each monomial x^j in c
3 : $k = k + 1$
4 : $\boldsymbol{z}_k = R_{\boldsymbol{s}x^j}(\boldsymbol{z}_{k-1})$
5 : **if** $\boldsymbol{z}_k = \perp$
6 : **return** $\perp$
7 : $r \leftarrow \mathcal{U}([0, 1))$
8 : **if** $r > M^{k-\kappa}$
9 : **return** $\perp$
10 : **return** $\boldsymbol{z}_k$

Fig. 3. Key-generation, signing and verification for an uncompressed version of our new scheme with an auxiliary function for rejection sampling.

have fewer than κ non-zero coefficients are therefore rejected with a suitable probability.

Bounds For our scheme, a discrete Gaussian distribution $\mathcal{D}_{\mathcal{R},\sigma}$ with parameter σ is used for sampling secrets. To bound the rejection probability, we ensure that $\|\boldsymbol{s}\| < B_{\mathrm{k}}$ by rejecting secret keys with norm larger than this. We select B_{k} as $\sigma\sqrt{n(m + \ell + 4)}$, resulting in approximately half of the secret keys being rejected.

Limiting the norm of the secret key by B_{k} results in secrets sampled from a narrower distribution, and should somewhat decrease the security of the scheme. However, as around half of the secret keys are still accepted, the security against lattice attacks should not decrease by significantly more than 1 bit. For simplicity, we do not account for this small security loss in our security estimates.

As $\boldsymbol{z}$ follows a discrete Gaussian distribution, there is no upper bound on $\|\boldsymbol{z}\|$. However, in order to ensure that signature forgery is hard, only sufficiently short $\boldsymbol{z}$ can be accepted as valid signatures. As such, only signatures that satisfy $\|\boldsymbol{z}\| \leq B_{\mathrm{s}}$ are emitted. If a $\boldsymbol{z}$ with larger norm

is generated during a signing attempt, the signature is rejected and the signing of the message is retried.
The bound B_s must be set to not result in too many signature rejections, while still only accepting relatively short $\boldsymbol{z}$. We therefore set $B_s = \left\lceil 1.02r\sqrt{n(\ell+m+1)} \right\rceil$, which from testing results in rejecting around 20% of constructed signatures. For the parametrizations targeting higher security levels, this rejection probability is somewhat lower. Note that this rejection of $\boldsymbol{z}$ with too large norm only applies to the final signature, and not to the intermediate $\boldsymbol{z}_i$ during the iterative rejection sampling.

4.2 More Compact Signatures

Compressing the signatures and encoding them in a compact way leads to significantly smaller signatures. The compression and encoding we use is very similar to that used for HAETAE, which in turn is similar to what many other lattice-based signature schemes use.
In particular, as with several recent schemes such as HAETAE and G+G, we make use of the rANS encoding of Duda [18] to encode our signatures. This results in signature sizes that are close to the entropy of the distribution that the signatures follow.
Additionally, we do not include the last m coordinates of the $\boldsymbol{z}$ vector in the signature, instead replacing them with a hint vector. This is similar to several other Fiat-Shamir signature schemes, including Dilithium [26]. Compressing the signatures in this way is enabled by not committing to the full $\boldsymbol{w}$ vector during signing, instead only committing to higher order bits of this vector. Due to this, the security of the scheme is based on the assumed hardness of the MSIS problem with a bound $B_v > B_s$. Analysis of the signature compression is relatively standard and we therefore omit this analysis from here and instead include it in Appendix A. This appendix also explain relevant details of the rANS encoding used.
An additional size optimization that is almost always used for lattice-based cryptosystems is to pseudorandomly derive uniformly random values from a short seed. In our case, this allows the verification key to be represented by $\boldsymbol{b}$ and a short seed from which $\boldsymbol{A}_0$ is derived. For the concrete verification key sizes we present, we therefore account for the size of $\boldsymbol{A}_0$ only as a 32-byte seed.

4.3 Security

The theoretical security of our signature scheme is given by Theorem 2 and follows from relatively standard arguments based on the Fiat-Shamir transform. In particular, the proof makes use of the fact that the produced signatures follow a distribution that is independent of the secret key of the scheme. As the proof follows the same structure as the one for HAETAE [11, 10] without much novelty, we include it separately in Appendix C.

Theorem 2. *Let γ be the commitment min-entropy of our signature scheme and β the probability that it restarts during signing. Furthermore, let $\mathcal{A}$ be a UF-CMA attacker against our signature scheme that makes q_S queries to the signing oracle and q_H queries to the random oracle $\mathcal{H}$. Then there are adversaries $\mathcal{B}$ and $\mathcal{B}'$ with essentially the same running time as $\mathcal{A}$ such that*

$$\begin{aligned}\mathsf{Adv}^{\textit{UF-CMA}}(\mathcal{A}) \leq \mathsf{Adv}^{NTWE}_{q,m,\ell,\mathcal{D}_{\mathcal{R},\sigma}}(\mathcal{B}) + \mathsf{Adv}^{\textit{BimodalSelfTargetMSIS}}_{\mathcal{H},q,m,\ell,B_v}(\mathcal{B}') \\ + \frac{2q_S 2^{-\gamma/2}}{1-\beta}\sqrt{q_H + 1 + \frac{q_S}{1-\beta}} + 2^{-\gamma/2+1}(q_H+1)\sqrt{\frac{q_S}{1-\beta}}.\end{aligned}$$

Additionally, let $\mathcal{A}'$ be an adversary against the sUF-CMA security of our signature scheme that makes q'_S queries to the signing oracle and q'_H queries to the random oracle $\mathcal{H}$. Then there are adversaries $\mathcal{F}, \mathcal{F}', \mathcal{F}''$ with essentially the same running time as $\mathcal{A}$ such that

$$\begin{aligned}\mathsf{Adv}^{\textit{sUF-CMA}}(\mathcal{A}') \leq 2\mathsf{Adv}^{NTWE}_{q,m,\ell,\mathcal{D}_{\mathcal{R},\sigma}}(\mathcal{F}) + \mathsf{Adv}^{\textit{BimodalSelfTargetMSIS}}_{\mathcal{H},q,m,\ell,B_v}(\mathcal{F}') \\ + \mathsf{Adv}^{MSIS}_{q,m,\ell,2B_v}(\mathcal{F}'') + \frac{2q'_S 2^{-\gamma/2}}{1-\beta}\sqrt{q'_H + 1 + \frac{q'_S}{1-\beta}} \\ + 2^{-\gamma/2+1}(q'_H+1)\sqrt{\frac{q'_S}{1-\beta}}.\end{aligned}$$

5 Concrete Parameters

In Table 1 we present some suggested parametrizations of our scheme which have a noticeable, but acceptable, rejection probability. Additionally, in Table 2 we present some parametrizations with negligible rejection probability. By applying some additional restrictions to the scheme, all rejection conditions of these parametrizations can be securely ignored. These additional restrictions are compatible with the parametrizations in Table 2 and are described in Appendix B. We also provide a comparison of the verification key and signature sizes of our scheme to that of alternative schemes in Table 3.

The public key size presented in the tables is

$$\left\lceil \frac{nm\lceil \log(q) \rceil}{8} \right\rceil + 32$$

bytes. This corresponds to representing each coefficient of $\boldsymbol{b}$ as a $\lceil \log(q) \rceil$ bit integer, while $\boldsymbol{A}_0$ is represented by a 32-byte seed. The concrete signature size we provide for our scheme is an approximation of the entropy for the distributions of compressed signatures plus 64 bytes. This includes 32 bytes for the representation of c and 32 additional bytes as leeway for padding to ensure a fixed signature size. For additional details on how signature size is estimated, see Appendix A.

We have not extensively investigated the probability that constructed signatures can be encoded into the size we present in the tables. However, from testing we have determined that at least a large majority of all signatures can be encoded to the presented size for all parametrizations that we consider.

5.1 Selecting Parameters

The security estimates of our proposed parametrizations presented in Tables 1 and 2 are computed with the lattice estimator [1]. For reproducibility, the concrete parameters used with the lattice estimator for our security estimates are presented in Appendix E. The security estimates are given via the

```
*.estimate.rough
```

functions, providing an approximation of the core-SVP security.
The security of our scheme follows from Theorem 2 and our security estimates follows from estimating the concrete hardness of the relevant variants of the BimodalSelfTargetMSIS and NTWE problems. For both of these underlying problems, we make the assumption that the structure present in $\mathcal{R}$ can not be efficiently exploited, and that the structured lattice problems are as hard as unstructured counterparts. Additionally, for the concrete security of the scheme, similar to HAETAE, we assume that the min-entropy of the scheme is large enough for us to safely ignore the contribution of these terms in Theorem 2. We include some additional notes regarding the min-entropy of the scheme in Appendix C.3.
The concrete parameters proposed in Tables 1 and 2 have been selected such that the estimated core-SVP security closely corresponds to the desired security levels. Additionally, the parameters are selected to have an acceptable rejection probability and in an attempt to minimize the combined size of a verification key and a signature. Besides parameters previously described, the tables also include the parameter τ that is related to the compression of the scheme, see Appendix A for details.

MSIS Security A classical reduction in the random oracle model shows that an algorithm solving the $\text{BimodalSelfTargetMSIS}_{\mathcal{H},q,m,\ell,B_{\text{v}}}$ problem efficiently can be used to efficiently solve the $\text{MSIS}_{q,m,\ell,2B_{\text{v}}}$ problem. However, this reduction is not tight and, as it makes use of rewinding and the forking lemma, it can not easily be translated to a reduction in the quantum random oracle model.
Instead of using this reduction to argue for the security of our scheme, we heuristically assume that the $\text{BimodalSelfTargetMSIS}_{\mathcal{H},q,m,\ell,B_{\text{v}}}$ problem is as hard as the $\text{MSIS}_{q,m,\ell,B_{\text{v}}}$ problem. Motivation for this approach is that the only way to solve this BimodalSelfTargetMSIS problem seems to be to to solve this corresponding MSIS problem. This same approach is also used in several previous works, such as Dilithium [26] and HAETAE [11, 10].

NTWE Security We estimate the hardness of the NTWE problem in the same way as in [20]. This assumes that the NTWE problem with $\boldsymbol{A} \in \mathcal{R}^{m \times \ell}$ is essentially as hard as the rank $\ell + 1$ module-LWE problem. More specifically, the $\text{NTWE}_{q,m,\ell,\mathcal{D}_{\mathcal{R},\sigma}}$ problem is estimated to be as hard to solve as the unstructured LWE problem with $\boldsymbol{A} \in \mathbb{Z}_q^{m' \times \ell'}$ where

Security	120	180	260
q	12289	50177	50177
ℓ	1	3	4
m	2	2	3
σ	2.60	1.00	1.50
r	128	55	95
κ	58	80	128
τ	256	128	128
B_{k}	110.07	48.00	79.60
B_{s}	4178	2199	4386
B_{v}	5649	2946	5300
Rejection rate	0.460	0.642	0.617
Rejection rate per coefficient	0.0106	0.0127	0.0075
Verification Key Size (bytes)	928	1056	1568
Signature Size (bytes)	775	1184	1694
Combined (bytes)	1703	2240	3262
BKZ block size to break SIS (Strong UF)	420 (332)	629 (500)	899 (730)
Core-SVP cost	122 (96)	183 (146)	262 (213)
BKZ block size to break NTWE	414	618	881
Core-SVP cost	120	180	257

Table 1. Proposed parametrizations of our signature scheme. The reported rejection rates only accounts for the probability that $\boldsymbol{z} = \perp$ is produced by RejectSample$(\boldsymbol{y}, c)$ and not the probability that $\|\boldsymbol{z}\| > B_{\mathrm{s}}$.

Security	120	180	260
q	50177	50177	50177
ℓ	2	3	4
m	2	3	4
σ	0.90	1.00	1.45
r	165	205	325
κ	58	80	128
τ	512	512	256
B_{k}	40.73	50.60	80.37
B_{s}	7541	10679	18769
B_{v}	10460	14254	20849
Rejection rate	2^{-111}	2^{-112}	2^{-111}
Verification Key Size (bytes)	1056	1568	2080
Signature Size (bytes)	1059	1475	2161
Combined (bytes)	2115	3043	4241
BKZ block size to break SIS (Strong UF)	416 (337)	670 (552)	889 (742)
Core-SVP cost	121 (98)	195 (161)	259 (216)
BKZ block size to break NTWE	422	618	874
Core-SVP cost	123	180	255

Table 2. Alternative parametrizations of our signature scheme which have a negligible rejection probability, allowing producing signatures without rejection.

$m' = mn$ and $\ell' = (\ell + 1)n - 1$. Furthermore, for the unstructured LWE problem, it is assumed that all elements of the secret are sampled from $\mathcal{D}_{\mathbb{Z},\sigma}$. With this estimate, as $f = 2f_0 + 1$ is deterministically constructed from $f_0 \leftarrow \mathcal{D}_{\mathcal{R},\sigma}$, we do not account for the fact that f is larger than if directly sampled from $\mathcal{D}_{\mathcal{R},\sigma}$. For our proposed parameters, the concrete estimated NTWE hardness is given by the estimated core-SVP hardness of solving such a search LWE problem.

5.2 Comparison to Other Schemes

Compactness In Table 3 we present the sizes of verification keys and signatures for our scheme and for other lattice-based signature schemes. Out of the presented signature schemes, only Falcon [28] and HAWK [8] have comparable sizes to our new system. Both HAWK and Falcon are based on the hash-and-sign paradigm using NTRU lattices. Meanwhile, the other schemes in the comparison are based on some variant of the Fiat-Shamir transform.
As can be seen in Table 3, our scheme is significantly more compact than all other Fiat-Shamir-based schemes in the comparison. Furthermore, even when parametrized to have negligible rejection probability, our scheme is still more compact than previous Fiat-Shamir-based schemes. The smaller verification keys and signatures are the main advantage of our new signature scheme compared to other Fiat-Shamir-based schemes.

Secure Implementation The more compact hash-and-sign-based schemes Falcon and HAWK have some downsides compared to the Fiat-Shamir-based schemes, and our scheme also has similar downsides. Compared to Dilithium, Falcon has the downside of implementations requiring floating point operations, which can be hard to implement in a side-channel secure manner.
Our scheme may also be hard to implement in a side-channel secure manner, as it has previously been observed that implementations of rejection sampling from Gaussian distributions may be vulnerable to side-channel attacks [9, 27]. Furthermore, our new more advanced rejection sampling may further complicate developing side-channel secure implementations. Although developing a secure implementation for our scheme almost certainly is harder than for schemes such as Dilithium and HAETAE, it is not directly clear how this aspect of our scheme compares to Falcon.

Hardness Assumptions Compared to other lattice-based schemes, HAWK has the downside of relying on the assumed hardness of a problem that is not as well studied as other lattice problems. Our scheme has the similar downside on relying on the relatively new NTWE problem. However, as the NTWE problem can be seen as a natural combination of the NTRU and LWE problems, it can be argued to be better understood than the lattice isomorphism problem upon which HAWK is based.
Furthermore, as explained in Section 1.3, our scheme could also have been based on NTRU or MLWE lattices. An NTRU-based variant of our

scheme would have similar sizes as the one we propose, but with less flexibility in how parameters can be selected. Meanwhile, an MLWE-based variant of our scheme would have the same flexibility in parameter choices, but with somewhat larger sizes. Our NTWE-based parametrization can be directly transferred to an MLWE-based variant of the scheme, leading to a scheme with similar security that has the same verification key sizes but with signatures that are 250–300 bytes larger for all targeted security levels. Thus, an MLWE-based variant of our scheme would still be significantly more compact than previous lattice-based Fiat-Shamir signature schemes.

Efficiency Another point of comparison is how long it takes to sign a message. Due to the more advanced rejection sampling of our scheme, signing may be somewhat slower with our scheme than with previous Fiat-Shamir-based signature schemes. Additionally, our signing must repeat the rejection sampling multiple times in an iterative manner in order to actually produce a signature, increasing the runtime of signing.
We do, however, believe that each iterative step of the rejection sampling should have a cost comparable to that of previous approaches. Although the functions that should be computed are expressed as an infinite sum, Lemma 2 shows that it is sufficient to account only for a few terms of the sum. Additionally, the lemma shows that the sum only depends on the two integers $\langle \boldsymbol{y}, \boldsymbol{v} \rangle$ and $\|\boldsymbol{v}\|^2$ and only minimal computations that handle high dimensional vectors are thus required to compute the sum.
As we have not developed an optimized implementation of our scheme, we can not fairly compare the execution time of our scheme to that of previous schemes. However, we believe that in many contexts, the smaller signatures and verification keys of our scheme more than make up for the somewhat longer signing time.

6 Conclusion

Thanks to our new rejection sampling, we are able to construct a signature scheme that is significantly more compact than previous lattice-based Fiat-Shamir signature schemes. There are, however, downsides to our new scheme and still potential for it to be further improved.
A notable downside with our new sampling is its reliance on discrete Gaussian distributions which may be hard to implement in a side-channel secure manner. Meanwhile, uniformly random distributions over hypercubes and hyperballs can be sampled from relatively easily, motivating their use for Dilithium and HAETAE respectively. It may therefore be interesting if our new approach to rejection sampling could be adapted to distributions that are easier to sample from.
Improvements of our rejection sampling may also be possible by further using the available flexibility in how the $\boldsymbol{z}$ vector in signatures can be constructed. In the same way as for the G+G signature scheme, $\boldsymbol{z}$ can be constructed as $\boldsymbol{y}+\boldsymbol{s}c+2\boldsymbol{s}k$ for arbitrary $k \in \mathcal{R}$. We only use this flexibility to iteratively select the sign of coefficients of c, but further improvements

Scheme	Security Level	VK Size	Signature Size	Total
NTRU+Sign-512	93	768	751	1519
Falcon-512	120	897	666	1563
HAWK-512	120	1024	555	1579
Ours with rejection	120	928	775	1703
BLISS-512	87	896	831	1727
NTRU G+G-512	85	1021	992	2013
Ours without rejection	121	1056	1059	2115
HAETAE-120	119	992	1474	2466
G+G-120	121	1472	1677	3149
Dilithium-2	123	1312	2420	3732
Ours with rejection	180	1056	1184	2240
Ours without rejection	180	1568	1475	3043
NTRU+Sign-1024	211	1664	1551	3215
BLISS-1024	178	1792	1836	3628
HAETAE-180	180	1472	2349	3821
NTRU G+G-180	178	2080	1769	3849
G+G-180	178	1952	2143	4095
Dilithium-3	183	1952	3293	5245
Falcon-1024	273	1793	1280	3073
Ours with rejection	257	1568	1694	3262
HAWK-1024	260	2440	1221	3661
Ours without rejection	255	2080	2161	4241
HAETAE-260	256	2080	2948	5028
G+G-260	260	2336	2804	5140
Dilithium-5	252	2592	4595	7187

Table 3. Signature sizes in bytes for different lattice-based signature schemes. The sizes for BLISS are based on the updated analysis of [30]. Security level presented for HAWK is based directly on target NIST security category.

may be possible by not imposing this limitation in how k is selected. However, selecting k more freely would complicate the analysis needed to ensure that the sampling produces the correct output distribution.

Finally, techniques used for NTRU+Sign [30] may also be beneficial for our scheme in order to further decrease the signature sizes. In particular, by constructing the public key somewhat differently in NTRU+Sign, the bimodal rejection sampling can be performed modulo q instead of modulo $2q$ and a similar optimization may be possible with our scheme.

Acknowledgments. This research has been supported in part by the Swedish Armed Forces and was conducted at KTH Center for Cyber Defense and Information Security (CDIS). The author would like to thank Johan Håstad for extensive comments and useful discussions on early versions of this manuscript. The author also thanks Martin Ekerå for helpful feedback and comments.

References

1. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9**(3), 169–203 (2015). https://doi.org/doi:10.1515/jmc-2015-0016, https://doi.org/10.1515/jmc-2015-0016
2. Bai, S., Beard, A., Johnson, F., Vidhanalage, S.K.B., Ngo, T.: Fiat-shamir signatures based on module-NTRU. In: Nguyen, K., Yang, G., Guo, F., Susilo, W. (eds.) ACISP 22: 27th Australasian Conference on Information Security and Privacy. Lecture Notes in Computer Science, vol. 13494, pp. 289–308. Springer, Cham, Switzerland, Wollongong, NSW, Australia (Nov 28–30, 2022). https://doi.org/10.1007/978-3-031-22301-3˙15
3. Bai, S., Galbraith, S.D.: An improved compression technique for signatures based on learning with errors. In: Benaloh, J. (ed.) Topics in Cryptology – CT-RSA 2014. Lecture Notes in Computer Science, vol. 8366, pp. 28–47. Springer, Cham, Switzerland, San Francisco, CA, USA (Feb 25–28, 2014). https://doi.org/10.1007/978-3-319-04852-9˙2
4. Bai, S., Jangir, H., Lin, H., Ngo, T., Wen, W., Zheng, J.: Compact encryption based on module-NTRU problems. In: Saarinen, M.J., Smith-Tone, D. (eds.) Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part I. pp. 371–405. Springer, Cham, Switzerland, Oxford, UK (Jun 12–14, 2024). https://doi.org/10.1007/978-3-031-62743-9˙13
5. Banaszczyk, W.: New bounds in some transference theorems in the geometry of numbers. Mathematische Annalen **296**(1), 625–635 (1993). https://doi.org/10.1007/BF01445125, https://doi.org/10.1007/BF01445125
6. Barbosa, M., Barthe, G., Doczkal, C., Don, J., Fehr, S., Grégoire, B., Huang, Y.H., Hülsing, A., Lee, Y., Wu, X.: Fixing and mechanizing the security proof of Fiat-Shamir with aborts and Dilithium. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023, Part V. Lecture Notes in Computer Science, vol. 14085, pp. 358–389. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2023). https://doi.org/10.1007/978-3-031-38554-4˙12
7. Behnia, R., Chen, Y., Masny, D.: On removing rejection conditions in practical lattice-based signatures. In: Cheon, J.H., Tillich, J.P. (eds.) Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021. pp. 380–398. Springer, Cham, Switzerland, Daejeon, South Korea (Jul 20–22, 2021). https://doi.org/10.1007/978-3-030-81293-5˙20
8. Bos, J.W., Bronchain, O., Ducas, L., Fehr, S., Huang, Y., Pornin, T., Postlethwaite, E.W., Prest, T., Pulles, L.N., van Woerden, W.: HAWK. Tech. rep., National Institute of Standards and Technology (2023), available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures
9. Bruinderink, L.G., Hülsing, A., Lange, T., Yarom, Y.: Flush, gauss, and reload - A cache attack on the BLISS lattice-based signature

scheme. In: Gierlichs, B., Poschmann, A.Y. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2016. Lecture Notes in Computer Science, vol. 9813, pp. 323–345. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–19, 2016). https://doi.org/10.1007/978-3-662-53140-2˙16

10. Cheon, J.H., Choe, H., Devevey, J., Güneysu, T., Hong, D., Krausz, M., Land, G., Shin, J., Stehlé, D., Yi, M.: HAETAE. Tech. rep., National Institute of Standards and Technology (2023), available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures
11. Cheon, J.H., Choe, H., Devevey, J., Güneysu, T., Hong, D., Krausz, M., Land, G., Möller, M., Stehlé, D., Yi, M.: HAETAE: Shorter lattice-based Fiat-Shamir signatures. IACR Transactions on Cryptographic Hardware and Embedded Systems **2024**(3), 25–75 (Jul 2024). https://doi.org/10.46586/tches.v2024.i3.25-75, https://tches.iacr.org/index.php/TCHES/article/view/11669
12. del Pino, R., Espitau, T., Katsumata, S., Maller, M., Mouhartem, F., Prest, T., Rossi, M., Saarinen, M.: Raccoon. Tech. rep., National Institute of Standards and Technology (2023), available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures
13. Devevey, J., Fawzi, O., Passelègue, A., Stehlé, D.: On rejection sampling in Lyubashevsky's signature scheme. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology – ASIACRYPT 2022, Part IV. Lecture Notes in Computer Science, vol. 13794, pp. 34–64. Springer, Cham, Switzerland, Taipei, Taiwan (Dec 5–9, 2022). https://doi.org/10.1007/978-3-031-22972-5˙2
14. Devevey, J., Passelègue, A., Stehlé, D.: G+G: A fiat-shamir lattice signature based on convolved gaussians. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology – ASIACRYPT 2023, Part VII. Lecture Notes in Computer Science, vol. 14444, pp. 37–64. Springer, Singapore, Singapore, Guangzhou, China (Dec 4–8, 2023). https://doi.org/10.1007/978-981-99-8739-9˙2
15. Ducas, L.: Accelerating bliss: the geometry of ternary polynomials. Cryptology ePrint Archive, Report 2014/874 (2014), https://eprint.iacr.org/2014/874
16. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal Gaussians. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013, Part I. Lecture Notes in Computer Science, vol. 8042, pp. 40–56. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013). https://doi.org/10.1007/978-3-642-40041-4˙3
17. Ducas, L., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS – Dilithium: Digital signatures from module lattices. Cryptology ePrint Archive, Report 2017/633 (2017), https://eprint.iacr.org/2017/633
18. Duda, J.: Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding (2014), https://arxiv.org/abs/1311.2540

19. Espitau, T., Tibouchi, M., Wallet, A., Yu, Y.: Shorter hash-and-sign lattice-based signatures. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022, Part II. Lecture Notes in Computer Science, vol. 13508, pp. 245–275. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 15–18, 2022). https://doi.org/10.1007/978-3-031-15979-4˙9
20. Gärtner, J.: NTWE: A natural combination of NTRU and LWE. In: Johansson, T., Smith-Tone, D. (eds.) Post-Quantum Cryptography - 14th International Workshop, PQCrypto 2023. pp. 321–353. Springer, Cham, Switzerland, College Park, USA (Aug 16–18, 2023). https://doi.org/10.1007/978-3-031-40003-2˙12
21. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th Annual ACM Symposium on Theory of Computing. pp. 197–206. ACM Press, Victoria, BC, Canada (May 17–20, 2008). https://doi.org/10.1145/1374376.1374407
22. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Third Algorithmic Number Theory Symposium (ANTS). Lecture Notes in Computer Science, vol. 1423, pp. 267–288. Springer (Jun 1998)
23. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography **75**(3), 565–599 (2015). https://doi.org/10.1007/s10623-014-9938-4
24. Lyubashevsky, V.: Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) Advances in Cryptology – ASIACRYPT 2009. Lecture Notes in Computer Science, vol. 5912, pp. 598–616. Springer Berlin Heidelberg, Germany, Tokyo, Japan (Dec 6–10, 2009). https://doi.org/10.1007/978-3-642-10366-7˙35
25. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. Lecture Notes in Computer Science, vol. 7237, pp. 738–755. Springer Berlin Heidelberg, Germany, Cambridge, UK (Apr 15–19, 2012). https://doi.org/10.1007/978-3-642-29011-4˙43
26. Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehlé, D., Bai, S.: CRYSTALS-DILITHIUM. Tech. rep., National Institute of Standards and Technology (2022), available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`
27. Pessl, P.: Analyzing the shuffling side-channel countermeasure for lattice-based signatures. In: Dunkelman, O., Sanadhya, S.K. (eds.) Progress in Cryptology - INDOCRYPT 2016: 17th International Conference in Cryptology in India. Lecture Notes in Computer Science, vol. 10095, pp. 153–170. Springer, Cham, Switzerland, Kolkata, India (Dec 11–14, 2016). https://doi.org/10.1007/978-3-319-49890-4˙9
28. Prest, T., Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: FALCON. Tech. rep., National Institute of Standards and

Technology (2022), available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
29. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology **4**(3), 161–174 (Jan 1991). https://doi.org/10.1007/BF00196725
30. Woo, J., Kim, J., Hong, G.H., Lee, S., Kim, M., Lee, H., Park, J.H.: NTRU+sign: Compact NTRU-based signatures using bimodal distributions. Cryptology ePrint Archive, Paper 2025/106 (2025), https://eprint.iacr.org/2025/106

A More Compact Signatures

We present a compressed variant of our signature scheme in Figure 4. In Section A.1 we describe the signature compression used for this scheme, and in Section A.2 we describe how the resulting signatures can be encoded in a compact way.

The technique we use for signature compression somewhat alters the security assumptions for the scheme and results in the security of the scheme relying on a somewhat easier MSIS problem. Meanwhile, the choice of signature encoding has no security implications and, besides decreasing signature sizes, only impacts details in how the scheme is implemented.

CSign($\boldsymbol{A}, \boldsymbol{s}, \mu$)

1 : $\boldsymbol{y} \leftarrow \mathcal{D}^{\ell+m+1}_{\mathcal{R},r}$
2 : $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{y} \bmod 2q$
3 : $\boldsymbol{w}_0 = \mathrm{LSB}(\boldsymbol{w})$
4 : $c = \mathcal{H}(\mathrm{HighBits}(\boldsymbol{w}), \boldsymbol{w}_0, \mu)$
5 : $\boldsymbol{z} = \mathrm{RejectSample}(\boldsymbol{y}, c)$
6 : **if** $\boldsymbol{z} = \perp$ or $\|\boldsymbol{z}\| > B_\mathrm{s}$
7 : Goto line 1
8 : $(\boldsymbol{z}_1, \boldsymbol{z}_2) = \mathrm{Split}(\boldsymbol{z})$
9 : $\tilde{\boldsymbol{w}} = \boldsymbol{w} - 2\boldsymbol{z}_2 \bmod 2q$
10 : $\boldsymbol{h} = \mathrm{HighBits}(\boldsymbol{w}) - \mathrm{HighBits}(\tilde{\boldsymbol{w}}) \bmod 2(q-1)$
11 : **return** $(\boldsymbol{z}_1, \boldsymbol{h}, c)$

HighBits($\boldsymbol{w}$)

1 : **return** $\lfloor \boldsymbol{w}/\tau \rceil \cdot \tau \bmod 2(q-1)$

LowBits($\boldsymbol{w}$)

1 : **return** $\boldsymbol{w} - \mathrm{HighBits}(\boldsymbol{w}) \bmod^{\pm} 2q$

CVerify($(\boldsymbol{z}_1, \boldsymbol{h}, c), \boldsymbol{A}, \mu$)

1 : $\tilde{\boldsymbol{w}}' = \boldsymbol{A}_1\boldsymbol{z}_1 - qc\boldsymbol{j} \bmod 2q$
2 : $\boldsymbol{w}' = \mathrm{HighBits}(\tilde{\boldsymbol{w}}') + \boldsymbol{h} \bmod 2(q-1)$
3 : Let z_0 be the first element of $\boldsymbol{z}_1$
4 : $\boldsymbol{w}_0 = \mathrm{LSB}(z_0 - c)\boldsymbol{j}$
5 : $c' = \mathcal{H}(\boldsymbol{w}', \boldsymbol{w}_0, \mu)$
6 : $\boldsymbol{z}'_2 = (\boldsymbol{w}' - \tilde{\boldsymbol{w}}' + \boldsymbol{w}_0)/2 \bmod^{\pm} q$
7 : $\boldsymbol{z}' = [\boldsymbol{z}_1^T, \boldsymbol{z}_2'^T]^T$
8 : Accept if $c = c'$ and $\|\boldsymbol{z}'\| \leq B_\mathrm{v}$

Fig. 4. Compressed variant of signing algorithm and corresponding verification algorithm for our new scheme.

A.1 Signature Compression

For our scheme, we make use of $\boldsymbol{A} = [q\boldsymbol{j} - 2\boldsymbol{b}, 2\boldsymbol{A}_0, 2\boldsymbol{I}_m]$. This is a similar situation as the one in for instance HAETAE [11, 10], with the last m coordinates of $\boldsymbol{z}$ only having a small impact on the value of $\boldsymbol{Az}$. This leads to the idea of compressing the signature by not including these last m coordinates of $\boldsymbol{z}$ in the signature. To this end, we define the function Split($\boldsymbol{z}$) which splits $\boldsymbol{z} \in \mathcal{R}^{m+\ell+1}$ into $\boldsymbol{z}_1 \in \mathcal{R}^{\ell+1}$ and $\boldsymbol{z}_2 \in \mathcal{R}^m$, with $\boldsymbol{z} = [\boldsymbol{z}_1^T, \boldsymbol{z}_2^T]^T$.
With $\boldsymbol{z}_2$ not included in the signature, the verifier can no longer recompute $\boldsymbol{w}$. However, as these last coordinates only have a small impact, the verifier is still able to recompute a good approximation of the committed vector $\boldsymbol{w}$. To ensure that a signature without $\boldsymbol{z}_2$ actually is useful, the signer therefore does not commit to the full vector $\boldsymbol{w} = \boldsymbol{Ay} \bmod 2q$. Instead, the signer commits only to parts of $\boldsymbol{w}$ that are not heavily influenced by $\boldsymbol{z}_2$.
To ensure that the committed value can be correctly recomputed by the verifier, the signer also includes a small hint $\boldsymbol{h}$ with the signature. This hint describes the behaviour that $\boldsymbol{z}_2$ should have on the parts of $\boldsymbol{w}$ that the signer committed to, enabling the verifier to correctly recompute this part of $\boldsymbol{w}$. This is quite similar to the approach in Dilithium [26], and even more similar to the approach of a preliminary version [17] of Dilithium which made use of discrete Gaussian distributions.[5]
As with Dilitihum, we want to commit to high order bits of $\boldsymbol{w}$. To this end, we consider some integer τ that divides $2(q-1)$ and round each coefficient of $\boldsymbol{w}$ to the nearest multiple of τ. Even without knowledge of $\boldsymbol{z}_2$, the verifier can compute a good approximation $\tilde{\boldsymbol{w}}'$ of $\boldsymbol{w}$. Even though $\tilde{\boldsymbol{w}}'$ is a good approximation of $\boldsymbol{w}$, rounding the coefficients of $\tilde{\boldsymbol{w}}'$ to the nearest multiple of τ is not guaranteed to give the same result as doing the same with $\boldsymbol{w}$. Because of this, the signature also includes a hint vector $\boldsymbol{h}$ that consist of the number of multiples of τ that differ between the rounded values of corresponding coefficients in $\boldsymbol{w}$ and $\tilde{\boldsymbol{w}}'$.
In a similar manner as with Dilithium, we note that the difference modulo $2q$ between 0 and $2(q-1)$ is only 2. To somewhat decrease the size of the hint vectors at a minimal cost, we therefore consider 0 and $2(q-1)$ to be the same multiple of τ. This corresponds to first rounding coefficients to the nearest multiple of τ and then taking the result modulo $2(q-1)$.
For the security proof of the scheme, we must also commit to the least significant bits of $\boldsymbol{w}$. This is without extra cost, as the least significant bit of $\boldsymbol{Az} \bmod 2q$ is unaffected by $\boldsymbol{z}_2$ since the last m columns of $\boldsymbol{A}$ consist of $2\boldsymbol{I}_m$.

A.2 Signature Encoding

To encode our signatures, we follow [19] and use the entropy coding called range asymmetric numeral systems (rANS) by Duda [18]. We apply this coding method to the hint $\boldsymbol{h}$ and to $\boldsymbol{z}_1$. The size of the encoding of $\boldsymbol{h}$

[5] This preliminary version is available via `https://eprint.iacr.org/archive/2017/633/20170627:201152`

and $\boldsymbol{z}_1$ is close to the entropy of the distributions that $\boldsymbol{h}$ and $\boldsymbol{z}_1$ follow. To encode $\boldsymbol{c}$, we simply include an n-bit string with non-zero bits in the string corresponding to non-zero coefficients of $\boldsymbol{c}$.
For the coding of $\boldsymbol{z}_1$, we follow the same approach as [19] and split each coefficient of $\boldsymbol{z}_1$ into two parts. The tails, consisting of the $k = 2^{\lfloor \log(r) \rfloor}$ least significant bits of the coefficients, are included as is, without use of rANS encoding. Encoding the tail would not significantly benefit the scheme as the coefficient bits included in the tail are close to distributed uniformly at random. Meanwhile, the head part of $\boldsymbol{z}_1$, consisting of the coefficient bits not included in the tail, is encoded using rANS. Furthermore, the entire hint vector $\boldsymbol{h}$ is encoded using rANS.
The encoding of $\boldsymbol{h}$ and $\boldsymbol{z}_1$ does not result in a fixed output size. The size of the output is, however, quite close to the entropy of the distributions that these parts of the signature follow, and the variance of the size is not that large.
To encode the signatures, we determine a maximal acceptable size of the encoding of $\boldsymbol{h}$ and the head part of $\boldsymbol{z}_1$. With rANS, the combination of these are encoded into a single integer, and assuming this integer is not too large, it is padded to have exactly the maximal acceptable size. This leads to a constant size for the signatures, but also results in an additional rejection condition for the signature scheme, with the scheme retrying with a different $\boldsymbol{y}$ if the compressed signature is too large.
To determine the maximal size of signatures, the entropy of the distribution of $\boldsymbol{h}$ and $\boldsymbol{z}_1$ must be estimated. The entropy of $\boldsymbol{z}_1$ is determined by using that the entropy of $\mathcal{D}_{\mathbb{Z},r}$ is close to $\log(2\pi e r^2)/2$ when r is larger than the smoothing parameter for the integers. To estimate the entropy of $\boldsymbol{h}$, we assume that it follows a discrete Gaussian distribution with parameter $2r/\tau$, which is a good approximation when r is sufficiently large in relation to τ. Total entropy is thus estimated as

$$\frac{n(\ell+1)\log(2\pi e r^2) + nm\log(8\pi e(r/\tau)^2)}{2} \tag{8}$$

bits and we round up this value to an integer number of bytes.
We have not analyzed in detail how likely it is that signatures are larger than our selected fixed maximal size. However, for all parameters we propose we have verified that, at least a significant majority of the times, the signature can be encoded into the maximal accepted size.
Below, we present some additional details regarding how the signatures are encoded, and how rANS works.

rANS Encoding To encode our signatures, we make use of the rANS encoding of Duda [18]. This leads to signature sizes that are close to the entropy of the distribution that the signatures follow. As described in Section A.2, rANS is used to encode the high order bits of $\boldsymbol{z}$ and the full hint vector $\boldsymbol{h}$. The procedure in both cases is almost the same, but with the coding function parametrized differently, due to $\boldsymbol{h}$ and the high order bits of $\boldsymbol{z}$ following different distributions.
The encoding is parametrized by a precision 2^k, where we select $k = 16$ in our implementation that we have used to test sizes of signatures in

practice. We then define $f(s)$ as an integer approximately equal to $2^k \cdot p(s)$ where $p(s)$ is an estimate for the probability that an entry of the data to encode is given by the symbol s.
For the encoding of the higher order bits of $\boldsymbol{z}$, we determine $p(s)$ by summing over the different outcomes of a discrete Gaussian distribution that correspond to the rounded coefficient s. For the encoding of h we determine $f(s)$ similarly, but account for the coefficients of LowBits($\boldsymbol{w}$) by assuming that they are uniformly random. It is only the difference in $p(s)$ that differs between the encoding of the high bits of $\boldsymbol{z}_1$ and the encoding of $\boldsymbol{h}$.
Although not defined as such, we can think of the symbols to encode, corresponding to the coefficients of $\boldsymbol{h}$ and the high bits of $\boldsymbol{z}_1$, as sequential integers. This allows us to define $\mathrm{CDF}(s) = \sum_{v<s} f(s)$ and $s = \mathrm{symbol}(y)$ as the unique symbol s such that $\mathrm{CDF}(s) \leq y < \mathrm{CDF}(s+1)$.
The coding function $C(x, s)$ is defined via

$$C(x, s) = 2^k \cdot \left\lfloor \frac{x}{f(s)} \right\rfloor + (x \bmod (f(s))) + \mathrm{CDF}(s)$$

Meanwhile, the decoding function $D(x)$ first determines the symbol to decode as $s = \mathrm{symbol}(x \,\&\, (2^k - 1))$ where $\&$ is the bitwise AND operation. The decoding function is then defined via

$$D(x) = \left(f(s) \cdot \left\lfloor x \cdot 2^{-k} \right\rfloor + (x \,\&\, (2^k - 1)) - \mathrm{CDF}(s), s \right).$$

To encode a vector over $\mathcal{R}$, we begin with $x = x_0$ for arbitrary positive integer x_0 as initial value. A coefficient s of an entry in the vector is encoded via $x = C(x, s)$, thus updating the integer x. This continues for every coefficient of every entry of the vector, resulting in some large integer $x = x_e$ as the final output of the encoding.
Decoding begins with a large integer $x = x_e$, applies the decoding function to get (x, s) for a new x and a decoded symbol s. This continues until the expected number of symbols have been decoded, which results in $x = x_0$. This gives the same sequence of symbols that were encoded into x_e, although returned in reverse.
As the encoding can start from an arbitrary initial value x_0, we can encode both $\boldsymbol{h}$ and the high bits of $\boldsymbol{z}$ into a single value. This is accomplished by first encoding the high bits of $\boldsymbol{z}$ with $x_0 = 0$ to get some value $x = x_z$. Then, the hint is similarly encoded, but starting with $x = x_z$ resulting in the final state $x = x_{zh}$. To decode, the hint vector is first recovered via the decoding function starting from x_{zh}, with the final decoding function call resulting in $x = x_z$. The high bits of $\boldsymbol{z}$ can then be decoded from this value x_z, thus recovering all the encoded data.

A.3 Correctness

For the compressed variant of our signature scheme in Figure 4, it is not directly obvious that correctly generated signatures always pass verification. That this is the case follows from Lemma 5 below. To prove this lemma, we first prove the following property of the LowBits function.

Lemma 4. *Let $k > 0$ be an integer, q be a prime and τ an integer that divides $2(q-1)$. Then, it holds that*

$$\|\mathit{LowBits}(\boldsymbol{w} \bmod 2q)\|_\infty \leq \tau/2 + 2$$

for arbitrary $\boldsymbol{w} \in \mathcal{R}^k$.

Proof. Without loss of generality, we assume that $\boldsymbol{w} = \boldsymbol{w} \bmod 2q$ and thus $\|\boldsymbol{w}\|_\infty < 2q$. By the triangle inequality, we can then upper bound $\|\mathrm{LowBits}(\boldsymbol{w})\|_\infty = \big\|\boldsymbol{w} - \mathrm{HighBits}(\boldsymbol{w}) \bmod^{\pm} 2q\big\|_\infty$ by

$$\|\boldsymbol{w} - \lfloor \boldsymbol{w}/\tau \rceil \cdot \tau\|_\infty + \big\|\lfloor \boldsymbol{w}/\tau \rceil \cdot \tau - \mathrm{HighBits}(\boldsymbol{w}) \bmod^{\pm} 2q\big\|_\infty.$$

In this expression, the first term $\|\boldsymbol{w} - \lfloor \boldsymbol{w}/\tau \rceil \cdot \tau\|_\infty$ is at most $\tau/2$.
For the second term, we note that, as $\|\boldsymbol{w}\|_\infty < 2q$, $\mathrm{HighBits}(\boldsymbol{w})$ and $\lfloor \boldsymbol{w}/\tau \rceil \cdot \tau \bmod^{\pm} 2q$ differ only for coefficients c of an element of $\boldsymbol{w}$ such that $\lfloor c/\tau \rceil \cdot \tau = 2(q-1)$. For coordinates that do differ, the difference is between $-2 = 2(q-1) \bmod^{\pm} 2q$ and $0 = 2q \bmod^{\pm} 2q$, which is thus a difference of 2 modulo $2q$. As such, $\|\mathrm{LowBits}(\boldsymbol{w})\|_\infty \leq \tau/2 + 2$. □

Next, we prove that all generated signatures verify as expected. The proof of this lemma is very similar to a proof for the HAETAE signature scheme [11, 10], which proves an analogous statement.

Lemma 5. *Let $(\boldsymbol{A}, \boldsymbol{s}) \leftarrow \mathit{KeyGen}()$ and let $B_v \geq B_s + \sqrt{nm}(\tau/4 + 1)$. Then, for every message $\mu \in \mathcal{M}$ the verification*

$$\mathit{CVerify}(\mathit{CSign}(\boldsymbol{A}, \boldsymbol{s}, \mu), \boldsymbol{A}, \mu)$$

always succeeds.

Proof. For the signature $(\boldsymbol{z}_1, \boldsymbol{h}, c)$ to be accepted, it is required that the recomputed c' equals c and that the norm of the recomputed $\boldsymbol{z}'$ vector is sufficiently short.
The recomputed challenge c' and the original challenge c match if

$$\mathrm{HighBits}(\tilde{\boldsymbol{w}}') + \boldsymbol{h} \bmod 2q = \mathrm{HighBits}(\boldsymbol{A}\boldsymbol{y} \bmod 2q) \tag{9}$$

and $\mathrm{LSB}(\boldsymbol{A}\boldsymbol{y} \bmod 2q) = \mathrm{LSB}(z_0 - c)\boldsymbol{j}$. To see that (9) holds, note that

$$\boldsymbol{A}_1\boldsymbol{z}_1 \bmod 2q = \boldsymbol{A}\boldsymbol{z} - \boldsymbol{A}_2\boldsymbol{z}_2 \bmod 2q = \boldsymbol{A}\boldsymbol{y} + \boldsymbol{A}\boldsymbol{s}c - 2\boldsymbol{z}_2 \bmod 2q$$

and we thus have

$$\tilde{\boldsymbol{w}}' = \boldsymbol{A}_1\boldsymbol{z}_1 - qc\boldsymbol{j} \bmod 2q = \boldsymbol{w} - 2\boldsymbol{z}_2 \bmod 2q = \tilde{\boldsymbol{w}}$$

as $\boldsymbol{A}\boldsymbol{s} = q\boldsymbol{j}$. With

$$\boldsymbol{w}' = \mathrm{HighBits}(\tilde{\boldsymbol{w}}') + \boldsymbol{h} \bmod 2(q-1) = \mathrm{HighBits}(\tilde{\boldsymbol{w}}) + \boldsymbol{h} \bmod 2(q-1)$$

the definition of $\boldsymbol{h}$ directly gives us that $\mathrm{HighBits}(\boldsymbol{w}) = \boldsymbol{w}'$.
Next, note that $\boldsymbol{A} \bmod 2 = [\boldsymbol{j}, \boldsymbol{0}, \boldsymbol{0}]$ and thus, for every vector $\boldsymbol{x}$,

$$\mathrm{LSB}(\boldsymbol{A}\boldsymbol{x} \bmod 2q) = \mathrm{LSB}(\boldsymbol{A}\boldsymbol{x}) = x_0\boldsymbol{j} \bmod 2,$$

where x_0 is the first element of $\boldsymbol{x}$. Furthermore, $\boldsymbol{z} \equiv \boldsymbol{y} + \boldsymbol{s}c \mod 2$, and thus $\mathrm{LSB}(z_0) = y_0 + s_0 c \bmod 2$, where z_0, y_0 and s_0 are the first elements of $\boldsymbol{z}, \boldsymbol{y}$ and $\boldsymbol{s}$ respectively. As $s_0 = f = 2f + 1 \equiv 1 \bmod 2$, this gives that

$$\mathrm{LSB}(z_0 - c)\boldsymbol{j} = \mathrm{LSB}(y_0)\boldsymbol{j} = \mathrm{LSB}(\boldsymbol{A}\boldsymbol{y} \bmod 2q) = \mathrm{LSB}(\boldsymbol{w})$$

as desired.
Finally we show that, for correctly generated signatures, the recomputed $\boldsymbol{z}' = [\boldsymbol{z}_1^T, \boldsymbol{z}_2'^T]^T$ vector always has norm at most B_v. During signing, the size of $\boldsymbol{z} = [\boldsymbol{z}_1^T, \boldsymbol{z}_2^T]^T$ is bounded by $\|\boldsymbol{z}\| \leq B_\mathrm{s}$. We want to bound $\|\boldsymbol{z}_2 - \boldsymbol{z}_2'\|$ and to this end make use of the fact that

$$\boldsymbol{w} - \tilde{\boldsymbol{w}}' = \boldsymbol{w} - \tilde{\boldsymbol{w}} \equiv 2\boldsymbol{z}_2 \mod 2q.$$

Since $\boldsymbol{z}_2' = (\boldsymbol{w}' - \tilde{\boldsymbol{w}}' + \boldsymbol{w}_0)/2 \bmod^{\pm} q$, we thus have that

$$\boldsymbol{z}_2' = \boldsymbol{z}_2 + (\mathrm{LowBits}(\boldsymbol{w}) - \boldsymbol{w}_0)/2 \bmod^{\pm} q \tag{10}$$

by using that $\boldsymbol{w}' = \mathrm{HighBits}(\boldsymbol{w})$ and the definition of LowBits.
To bound $\|\boldsymbol{z}_2'\|$ we first note that, $\left\|\boldsymbol{x} \bmod^{\pm} q\right\| \leq \|\boldsymbol{x}\|$ for every vector $\boldsymbol{x}$. This gives that

$$\left\|\boldsymbol{z}_2'\right\| \leq \|\boldsymbol{z}_2\| + \|\mathrm{LowBits}(\boldsymbol{w}) + \boldsymbol{w}_0\|/2$$

and it follows that

$$\left\|\boldsymbol{z}'\right\| \leq \|\boldsymbol{z}\| + \sqrt{nm}(\tau/4 + 2) \leq B_\mathrm{v}$$

by using $\|\mathrm{LowBits}(\boldsymbol{w}) + \boldsymbol{w}_0\|^2 \leq nm\|\mathrm{LowBits}(\boldsymbol{w}) + \boldsymbol{w}_0\|_\infty^2$ which is then bounded by the triangle inequality with Lemma 4 and by using that $\|\boldsymbol{w}_0\|_\infty \leq 1 < 2$. As such, all correctly generated signatures satisfy that the recomputed $\boldsymbol{z}'$ has norm at most B_v and are thus accepted by the verification procedure. □

B Variant Without Rejection

Our signature scheme can be instantiated in such a way that the rejection conditions during signing can be safely ignored. The two different reasons for signatures being rejected are that the RejectSample procedure outputs $\boldsymbol{z} = \perp$ or that the produced vector $\boldsymbol{z} \neq \perp$ has too large norm. With sufficiently large r and B_s, the probability to trigger these rejection conditions is negligible.
The probability that RejectSample outputs $\boldsymbol{z} \neq \perp$ is $M^{-\kappa}$, and M can be bounded by using (6). In particular, this bound shows that if $r \geq 4B_\mathrm{k}$, then $1/M \geq 1 - 2^{-108}$. As $\kappa \leq 128 = 2^7$ for all parametrizations that we consider, with $r \geq 4B_\mathrm{k}$ the probability that RejectSample outputs $\perp$ is upper bounded by 2^{-100}.
Meanwhile, to ensure that the rejection condition $\|\boldsymbol{z}\| > B_\mathrm{s}$ is triggered only with a negligible probability, we set $B_\mathrm{s} = \left\lceil Cr\sqrt{n(\ell + m + 1)} \right\rceil$ for a sufficiently large constant C. This allows us to argue that the signatures produced do not have too large norm by using the following lemma of Banaszczyk.

Lemma 6 (Lemma 1.5 from [5]). *Let $C \geq 1$, $r > 0$ be reals and $n \geq 1$ be an integer. Then*

$$\Pr\left[\|\boldsymbol{z}\| > Cr\sqrt{n} \mid \boldsymbol{z} \leftarrow \mathcal{D}_{\mathbb{Z}^n,r}\right] \leq \left(C \cdot \exp\left(\frac{1-C^2}{2}\right)\right)^n$$

The probability that $\|\boldsymbol{z}\| > B_{\mathrm{s}}$ is thus at most $(Ce^{(1-C^2)/2})^{(n(\ell+m+1))}$. To ensure that the probability for the rejection condition $\boldsymbol{z} > B_{\mathrm{s}}$ is at most 2^{-128}, we require $C \in [1.2, 1.3]$ for our different parametrizations.
An additional reason for rejections when using rANS encoding of signatures is due to our requirement that signatures should be padded to a fixed size. Because of this, if a produced signature can not be encoded into this fixed size, it has to be rejected. This rejection condition can easily be removed by allowing variable size signatures. Variable size signatures additionally have the advantage that such signatures usually have smaller size than our determined fixed size.

C Theoretical Security

To prove the security of our signature scheme, we take the same approach as HAETAE [11, 10]. We thus consider a canonical identification (CID) scheme from which our signature scheme can be constructed by using the Fiat-Shamir transform. The UF-CMA and sUF-CMA security of our signature scheme then follows from the UF-NMA security of the scheme and properties of the CID scheme.
To start with, in Section C.1, we provide some background on the Fiat-Shamir transform, defining the CUR property and presenting Theorem 3 which is used to prove the security of our scheme.
Next, the UF-NMA security of our scheme is given by Lemma 7 in Section C.2. Next, various details of the underlying CID scheme are presented in Section C.3, including Lemma 8 which proves that the scheme has a certain zero-knowledge property. Combining these lemmas with Theorem 3 proves the UF-CMA security of our signature scheme and gives us the first part of Theorem 2.
Finally, Lemma 9 proves the CUR property for our signature scheme, from which the sUF-CMA security of our scheme follows via Theorem 3, providing the second part of Theorem 2. Additionally, the security of our scheme depends on its min-entropy, and this aspect of our scheme is briefly discussed in Section C.3.

C.1 Background on the Fiat-Shamir Transform

The Fiat-Shamir transform allows transforming an identification scheme into a signature scheme. The security of the resulting signature scheme requires that the underlying identification scheme has the following zero-knowledge property.

Definition 5 (paHVZK). *Let ID be a canonical identification scheme with the scheme producing some output distribution of $(\boldsymbol{w}, c, \boldsymbol{z})$. The*

scheme is Perfect Accepting Honest Verifier Zero-Knowledge (paHVZK) if there exist a probabilistic polynomial time simulator that, when given only the verification key of the identification scheme, outputs $(\boldsymbol{w}', c', \boldsymbol{z}')$ *that, conditioned on* $\boldsymbol{z} \neq \perp$*, follows the same distribution as the output* $(\boldsymbol{w}, c, \boldsymbol{z})$*.*

If the identification scheme is paHVZK, the UF-CMA security of the transformed signature scheme directly follows from its UF-NMA security, via Theorem 3 below. From this same theorem, the sUF-CMA security of our scheme also follows. However, for the sUF-CMA security, the underlying identification scheme must also have the following computationally unique response (CUR) property.

Definition 6 (Computationally Unique Response (CUR)). *Let ID be a canonical identification scheme with instance generator Gen and let* $(x, y) \leftarrow$ *Gen, with* x *the public part of the key for the prover. The advantage of an adversary* $\mathcal{A}$ *against the CUR property of the scheme is given by the probability that* $\mathcal{A}(x)$ *outputs a commitment* w*, a challenge* c *and two separate responses* z *and* z' *such that the verifier accepts both* (w, c, z) *and* (w, c, z') *as valid transcripts.*

The theorem we use to prove the UF-CMA and sUF-CMA security of our scheme is an adaptation of Theorem 2 of [6]. Our adaptation of the original theorem is very similar to that of HAETAE [11].

Theorem 3 (Adapted from Theorem 2 of [6]). *Let ID be an identification scheme with commitment min-entropy* γ *that satisfies paHVZK with probability of aborting* β*. Let* $\mathcal{A}$ *be a quantum UF-CMA attacker against the Fiat-Shamir transform of ID with random oracle* H*. Furthermore, let* q_S *and* q_H *be the number of queries* $\mathcal{A}$ *makes to the signing oracle and to the random oracle* H *respectively. Then, there exists a quantum NMA attacker* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{\textit{UF-CMA}}(\mathcal{A}) \leq \mathsf{Adv}^{\textit{UF-NMA}}(\mathcal{B}) + \frac{2q_S 2^{-\gamma/2}}{1-\beta}\sqrt{q_H + 1 + \frac{q_S}{1-\beta}} + 2^{-\gamma/2+1}(q_H+1)\sqrt{\frac{q_S}{1-\beta}}.$$

Furthermore, if $\mathcal{A}$ *is an adversary against the sUF-CMA security, there exist an adversary* $\mathcal{B}'$ *against the CUR property of ID such that the previous bound holds by adding* $\mathsf{Adv}^{CUR}_{ID}(\mathcal{B}')$ *on the right hand side. The running time of* $\mathcal{B}$ *and* $\mathcal{B}'$ *is approximately that of* $\mathcal{A}$ *plus* q_S *times the running time of the paHVZK simulator.*

C.2 UF-NMA Security

The UF-NMA security of the uncompressed version of our scheme is almost directly given by the assumed hardnesses of the decision NTWE problem and the BimodalSelfTargetMSIS problem. For the compressed version of our scheme, the UF-NMA security similarly relies on the assumed hardness of these problems, but not quite as obviously. For both versions of our scheme, we argue for their UF-NMA security by the following lemma.

Lemma 7. *Let $\mathcal{A}$ be an adversary against the UF-NMA security of our signature scheme. Then, there exist two adversaries $\mathcal{B}$ and $\mathcal{B}'$, each using $\mathcal{A}$ once and performing negligible additional work, such that*

$$\mathsf{Adv}^{\text{UF-NMA}}(\mathcal{A}) \leq \mathsf{Adv}^{NTWE}_{q,m,\ell,\mathcal{D}_{\mathcal{R},\sigma}}(\mathcal{B}) + \mathsf{Adv}^{BimodalSelfTargetMSIS}_{\mathcal{H},q,m,\ell,B_v}(\mathcal{B}').$$

Proof. For the proof, we consider a variant U of our scheme for which the $\boldsymbol{A}_0$ and $\boldsymbol{b}$ in the public key are uniformly random instead of given by an NTWE instance. If $\mathcal{A}$ behaves differently against U than against our actual scheme, this can be used to solve the decision NTWE problem. The adversary $\mathcal{B}$ is defined to exploit this potential difference in the behaviour of $\mathcal{A}$ by using an input decision NTWE instance to construct the verification key for $\mathcal{A}$. If a forgery is created, it determines the input to be an NTWE instances and otherwise it determines it to be uniformly random.
The algorithm $\mathcal{B}'$ instead relies on whatever advantage $\mathcal{A}$ has against the variant U of our scheme. To achieve this, the input to the target BimodalSelfTargetMSIS problem is used to construct the verification key that is provided to $\mathcal{A}$. This verification key is thus constructed from uniformly random $\boldsymbol{A}_0$ and $\boldsymbol{b}$. For the uncompressed version of our signature scheme, a forgery produced by $\mathcal{A}$ directly corresponds to a solution for the BimodalSelfTargetMSIS problem of $\mathcal{B}'$, which proves the statement for the uncompressed version.
For the compressed version of our signature scheme, the random oracle that $\mathcal{B}'$ provides when simulating $\mathcal{A}$ is given by

$$\mathcal{H}'(\boldsymbol{w}', w_0, \mu) = \mathcal{H}(\boldsymbol{w}' + w_0, \mu)$$

where $\mathcal{H}$ is the random oracle for the BimodalSelfTargetMSIS instance. A forgery from the signature scheme consists of a signature $(\boldsymbol{z}_1, \boldsymbol{h}, c)$ and a message μ such that

$$\mathcal{H}'(\boldsymbol{w}', \boldsymbol{w}_0, \mu) = \mathcal{H}(\boldsymbol{w}' + \boldsymbol{w}_0, \mu) = c$$

for $\boldsymbol{w}'$ and $\boldsymbol{w}_0$ defined as in $\text{CVerify}((\boldsymbol{z}_1, \boldsymbol{h}, c), \boldsymbol{A}, \mu)$.
A valid forgery additionally corresponds to a vector $\boldsymbol{z}' = [\boldsymbol{z}_1^T, \boldsymbol{z}_2'^T]^T$ such that $\|\boldsymbol{z}'\| \leq B_{\text{v}}$, where $\boldsymbol{z}_2' = (\boldsymbol{w}' + \boldsymbol{w}_0 - \tilde{\boldsymbol{w}}')/2$. Given such a forgery, we claim that $(\boldsymbol{z}', c, \mu)$ solves the target BimodalSelfTargetMSIS instance. This is the case as

$$\boldsymbol{A}\boldsymbol{z}' = \boldsymbol{A}_1\boldsymbol{z}_1 + 2\boldsymbol{z}_2' = \boldsymbol{w}' + \boldsymbol{w}_0 + qc\boldsymbol{j}$$

and thus

$$\mathcal{H}(\boldsymbol{A}\boldsymbol{z}' - qc\boldsymbol{j}, \mu) = \mathcal{H}(\boldsymbol{w}' + \boldsymbol{w}_0, \mu) = c$$

as claimed. Furthermore, valid signatures satisfy that $\|\boldsymbol{z}'\| \leq B_{\text{v}}$ and therefore the norm bound for the BimodalSelfTargetMSIS instance is satisfied. □

C.3 CID Properties

The CID scheme that corresponds to the uncompressed version of our scheme is illustrated in Figure 5, together with a simulator for the scheme.

The instance generator of the identification scheme is the key-generation algorithm of the signature scheme, which provides $(\boldsymbol{A}, \boldsymbol{s})$, and we therefore do not include a separate description of the instance generator. For simplicity, we also do not provide an explicit definition of the CID scheme corresponding to the compressed variant of our signature scheme, but we still prove properties about both variants. We do this by detailing how the proofs can be adapted from the uncompressed setting to the compressed setting.

CID scheme

$\mathsf{P}(\boldsymbol{A}, \boldsymbol{s})$		$\mathsf{V}(\boldsymbol{A})$
$\boldsymbol{y} \leftarrow \mathcal{D}_{\mathcal{R},r}^{\ell+m+1}$		
$\boldsymbol{w} = \boldsymbol{A}\boldsymbol{y} \bmod 2q$	$\xrightarrow{\boldsymbol{w}}$	$c \leftarrow \mathcal{U}(\mathcal{C})$
$\boldsymbol{z}_0 = \boldsymbol{y}$	$\xleftarrow{c}$	
for Each non-zero coefficient c_i of c		
$\quad \boldsymbol{z}_i = R_{\boldsymbol{s}c_i}(\boldsymbol{z}_{i-1})$		
$\quad$ **if** $\boldsymbol{z}_i = \perp$		
$\quad\quad$ Abort CID scheme		
$k = \|c\|_1$		
Abort with probability $1 - M^{k-\kappa}$		
$\boldsymbol{z} = \boldsymbol{z}_k$	$\xrightarrow{\boldsymbol{z}}$	Reject if $\|\boldsymbol{z}\| > B$
		Accept if $\boldsymbol{A}\boldsymbol{z} - qc \mod 2q = \boldsymbol{w}$

CID simulator

1 : $\boldsymbol{z} \leftarrow \mathcal{D}_{\mathcal{R},r}^{\ell+m+1}$
2 : $c \leftarrow \mathcal{U}(\mathcal{C})$
3 : $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{z} - qc$
4 : **return** $(\boldsymbol{z}, c, \boldsymbol{w})$

Fig. 5. Canonical identification scheme and simulator corresponding to the uncompressed version of our signature scheme.

Zero knowledge First, in the following lemma we show that the CID scheme is paHVZK. For the scheme to be paHVZK, it must be possible to simulate the output from the scheme conditioned on $\boldsymbol{z} \neq \perp$ without knowledge of the secret key. To simulate the CID scheme, Lemma 1 is used to show that the output of $R_{\boldsymbol{v}}(\boldsymbol{y})$ is independent of $\boldsymbol{v}$, and thus the distribution of $\boldsymbol{z} \neq \perp$ is independent of the secret key.

Lemma 8. *The CID schemes that corresponds to the compressed and uncompressed variants of our signature scheme are paHVZK. Furthermore, the prover produces a* $\boldsymbol{z} \neq \perp$ *with probability* $M^{-\kappa}$.

Proof. The CID scheme for the uncompressed variant of our signature scheme and a simulator of the CID scheme is given in Figure 5. Combining Lemma 1 and Corollary 1 gives that, conditioned on $\boldsymbol{z}_i \neq \perp$, $\boldsymbol{z}_i$ is distributed as if from $\mathcal{D}_{\mathcal{R},r}^{\ell+m+1}$ for every $i \in \{1, \ldots, k\}$. As such, conditioned on the prover producing an output, the distribution of $\boldsymbol{z}$ is the same in the simulator and the CID scheme.
As $R_{\boldsymbol{v}}(\boldsymbol{y})$ has a probability $1/M$ of producing a valid output, the probability that it successfully produces an output all k times it is executed is M^{-k}. After z_k has been produced, the scheme aborts with probability $1 - M^{k-\kappa}$, ensuring that the probability of an output $\boldsymbol{z} \neq \perp$ is $M^{-\kappa}$ for every possible challenge c. Therefore, the rejection sampling does not impact the distribution of challenges in transcripts where $\boldsymbol{z} \neq \perp$.
Conditioned on $\boldsymbol{z} \neq \perp$, for both the simulator and the CID scheme, the challenge c is distributed as if sampled from $\mathcal{U}(\mathcal{C})$. This also implies that $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{z} - qc\boldsymbol{j}$ is distributed identically in the simulator and the CID scheme. As such, the distribution of $\boldsymbol{z}, c, \boldsymbol{w}$ from the simulator is the same as the one from the CID scheme conditioned on $\boldsymbol{z} \neq \perp$.
In the CID scheme corresponding to the compressed signature scheme, the verifier would first send $\boldsymbol{w}_h = \text{HighBits}(\boldsymbol{w})$ and $\boldsymbol{w}_0 = \text{LSB}(\boldsymbol{w})$. The challenge c would still be the same, but the final response of the verifier would only contain parts of $\boldsymbol{z}$ together with a hint derived from $\boldsymbol{z}$ and $\boldsymbol{w}$. As such, the transcript from a compressed variant of the CID scheme is given directly by a deterministic transform of the transcript from the uncompressed version. The same transform can be applied to the simulated transcript, and thus this version of the CID scheme is also paHVZK. □

Commitment min-entropy Next, we note that the commitment min-entropy of the compressed scheme is easily lower-bounded by close to n bits. This is the case as $\boldsymbol{w}_0 = \text{LSB}(y_0)\boldsymbol{j}$ is part of the commitment with y_0 sampled from a discrete Gaussian distribution with relatively large standard deviation. As such, the coefficients of the first element of $\boldsymbol{w}_0$ are statistically close to uniformly random, and $\boldsymbol{w}_0$ therefore has a min-entropy that is not significantly smaller than n.
Note, however, that this should significantly underestimate the commitment min-entropy of the scheme, as it only accounts for the min-entropy in $\boldsymbol{w}_0$. Although we do not prove this, similarly to in HAETAE [11], we feel that it is safe to assume that the actual commitment min-entropy is significantly larger than this. We therefore ignore terms which depend on the min-entropy when we use Theorem 2. Furthermore, we note that the uncompressed variant of the scheme obviously has at least as high min-entropy as the compressed variant.

Computational Unique Response To prove the sUF-CMA security of the signature scheme, we must also bound the advantage an

adversary may have against the CUR property of the CID scheme. Such a bound is presented in the following lemma.

Lemma 9. *If $\mathcal{A}$ is an adversary against the CUR property of the CID scheme, then there exist adversaries $\mathcal{B}$ and $\mathcal{B}'$, each using $\mathcal{A}$ once and performing negligible additional work, such that*

$$\mathsf{Adv}^{CUR}(\mathcal{A}) \leq \mathsf{Adv}^{NTWE}_{q,m,\ell,\mathcal{D}_{\mathcal{R},\sigma}}(\mathcal{B}) + \mathsf{Adv}^{MSIS}_{q,m,\ell,2B_v}(\mathcal{B}').$$

Proof. A successful adversary $\mathcal{A}$ against the CUR property of the CID scheme produces $\boldsymbol{w}, c, \boldsymbol{z}_a, \boldsymbol{z}_b$ with $\boldsymbol{z}_a \neq \boldsymbol{z}_b \bmod 2q$ such that $\boldsymbol{w}, c, \boldsymbol{z}_a$ and $\boldsymbol{w}, c, \boldsymbol{z}_b$ both are accepting transcripts for the CID scheme. Thus,

$$\boldsymbol{A}\boldsymbol{z}_a - qc\boldsymbol{j} \bmod 2q = \boldsymbol{A}\boldsymbol{z}_b - qc\boldsymbol{j} \bmod 2q = \boldsymbol{w}$$

and therefore $\boldsymbol{A}(\boldsymbol{z}_a - \boldsymbol{z}_b) = \boldsymbol{0} \bmod 2q$. We can reduce this modulo q and multiply by the inverse of 2 modulo q, which gives

$$[-\boldsymbol{b}, \boldsymbol{A}_0, \boldsymbol{I}_m](\boldsymbol{z}_a - \boldsymbol{z}_b) = \boldsymbol{0} \bmod q$$

as $\boldsymbol{A} = [qc\boldsymbol{j} - 2\boldsymbol{b}, 2\boldsymbol{A}_0, 2\boldsymbol{I}_m]$. As such, if $\boldsymbol{b}$ was uniformly random, then $\boldsymbol{z}_a - \boldsymbol{z}_b$ would correspond to a solution to an MSIS instance.
In the scheme $\boldsymbol{b}$ is generated as an NTWE instance and is not uniformly random. However, if $\mathcal{A}$ works when $\boldsymbol{b}$ is given by an NTWE instance and not when it is uniformly random, then $\mathcal{A}$ provides a distinguisher for the decision NTWE problem. Thus, $\mathcal{B}$ is constructed by using its input decision NTWE instance as verification key for $\mathcal{A}$ whereas $\mathcal{B}'$ constructs a verification key for $\mathcal{A}$ by sampling both $\boldsymbol{b}$ and $\boldsymbol{A}_0$ uniformly at random. With $\|\boldsymbol{z}_a\|$ and $\|\boldsymbol{z}_b\|$ both at most B_v, this proves the lemma for the uncompressed variant of our scheme.
For a compressed variant of the CID scheme, the adversary $\mathcal{A}$ would instead produce $\boldsymbol{w}', c, \boldsymbol{z}_{a,1}, \boldsymbol{h}_a, \boldsymbol{z}_{b,1}, \boldsymbol{h}_b$, with $(\boldsymbol{z}_{a,1}, \boldsymbol{h}_a) \neq (\boldsymbol{z}_{b,1}, \boldsymbol{h}_b)$. Assuming that this guarantees that the reconstructed $\boldsymbol{z}'_a$ and $\boldsymbol{z}'_b$ differ, the same argument as for the uncompressed variant holds. As the reconstructed $\boldsymbol{z}'$ is given by $[\boldsymbol{z}_1^T, \boldsymbol{z}_2'^T]^T$, that $\boldsymbol{z}'_a = \boldsymbol{z}'_b$ implies that $\boldsymbol{z}_{a,1} = \boldsymbol{z}_{b,1}$. Additionally, with $\boldsymbol{z}_{a,1} = \boldsymbol{z}_{b,1}$, in order for $\boldsymbol{z}_{a,2} = \boldsymbol{z}_{b,2}$ it is required that $\boldsymbol{w}'_a \equiv \boldsymbol{w}'_b \bmod 2(q-1)$ and this is only possible if $\boldsymbol{h}_a \equiv \boldsymbol{h}_b \bmod 2(q-1)$. To ensure that there is a computationally unique response it should thus be verified that $\boldsymbol{h}$ is the expected representative modulo $2(q-1)$. With such a verification, it is guaranteed that two different verifying transcripts for the compressed CID scheme imply two different reconstructed $\boldsymbol{z}'_a$ and $\boldsymbol{z}'_b$, both of which have norm at most B_v. In the same way as for the uncompressed variant, this directly provides a solution to the MSIS variant with size bound $2B_\mathrm{v}$. □

D Missing Proofs

D.1 Proof of Lemma 2

Here, we prove Lemma 2. For reference, we begin by restating the lemma.

Lemma 10. *The function $S_{\boldsymbol{v}}(\boldsymbol{y})$ only depends on $\langle \boldsymbol{y}, \boldsymbol{v} \rangle$ and $\|\boldsymbol{v}\|^2$. Furthermore, for arbitrary $t \geq 1$, fixed $\boldsymbol{v}$ and with $\langle \boldsymbol{y}, \boldsymbol{v} \rangle \geq -\|\boldsymbol{v}\|^2$ then*

$$\left| S_{\boldsymbol{v}}(\boldsymbol{y}) - \sum_{k=0}^{t} (-1)^k \frac{\rho_r(\boldsymbol{y} + k\boldsymbol{v})}{\rho_r(\boldsymbol{y})} \right| \leq \frac{\rho_r(t\boldsymbol{v})}{\rho_r(\boldsymbol{v}) - \rho_r(2\boldsymbol{v})}$$

Proof. To see that $S_{\boldsymbol{v}}(\boldsymbol{y})$ only depends on $\langle \boldsymbol{y}, \boldsymbol{v} \rangle$ and $\|\boldsymbol{v}\|^2$, note that

$$\rho_r(\boldsymbol{y} + k\boldsymbol{v}) = \exp\left(-\frac{\|\boldsymbol{y}\|^2 + k^2\|\boldsymbol{v}\|^2 + 2k\langle \boldsymbol{y}, \boldsymbol{v} \rangle}{2r^2} \right).$$

As such, we have

$$S_{\boldsymbol{v}}(\boldsymbol{y}) = \sum_{k=0}^{\infty} (-1)^k \frac{\rho_r(\boldsymbol{y} + k\boldsymbol{v})}{\rho_r(\boldsymbol{y})} = \sum_{k=0}^{\infty} (-1)^k \exp\left(-\frac{k^2\|\boldsymbol{v}\|^2 + 2k\langle \boldsymbol{y}, \boldsymbol{v} \rangle}{2r^2} \right)$$

which is a function that depends only on $\|\boldsymbol{v}\|^2$ and $\langle \boldsymbol{y}, \boldsymbol{v} \rangle$.
Next, we note that

$$\begin{aligned}
\left| S_{\boldsymbol{v}}(\boldsymbol{y}) - \sum_{k=0}^{t} (-1)^k \frac{\rho_r(\boldsymbol{y} + k\boldsymbol{v})}{\rho_r(\boldsymbol{y})} \right| &= \left| \sum_{k=t+1}^{\infty} (-1)^k \frac{\rho_r(\boldsymbol{y} + k\boldsymbol{v})}{\rho_r(\boldsymbol{y})} \right| \\
&\leq \sum_{k=t+1}^{\infty} \frac{\rho_r(\boldsymbol{y} + k\boldsymbol{v})}{\rho_r(\boldsymbol{y})} = \sum_{k=t+1}^{\infty} \exp\left(-\frac{k^2\|\boldsymbol{v}\|^2 + 2k\langle \boldsymbol{y}, \boldsymbol{v} \rangle}{2r^2} \right)
\end{aligned}$$

and when $\langle \boldsymbol{y}, \boldsymbol{v} \rangle \geq -\|\boldsymbol{v}\|^2$ this is upper bounded by

$$\begin{aligned}
&\sum_{k=t+1}^{\infty} \exp\left(-\frac{(k^2 - 2k)\|\boldsymbol{v}\|^2}{2r^2} \right) = \sum_{k=t+1}^{\infty} \exp\left(-\frac{((k-1)^2 - 1)\|\boldsymbol{v}\|^2}{2r^2} \right) \\
&\quad = \exp\left(\frac{\|\boldsymbol{v}\|^2}{2r} \right) \sum_{k=t}^{\infty} \exp\left(-\frac{k^2\|\boldsymbol{v}\|^2}{2r^2} \right) \\
&\quad \leq \exp\left(-\frac{(t^2 - 1)\|\boldsymbol{v}\|^2}{2r^2} \right) \sum_{k=0}^{\infty} \exp\left(-\frac{k^2\|\boldsymbol{v}\|^2}{2r} \right) \\
&\quad \leq \exp\left(-\frac{(t^2 - 1)\|\boldsymbol{v}\|^2}{2r^2} \right) \sum_{k=0}^{\infty} \exp\left(-\frac{k\|\boldsymbol{v}\|^2}{2r^2} \right) \\
&\quad \leq \frac{\exp\left(-\frac{(t^2 - 1)\|\boldsymbol{v}\|^2}{2r^2} \right)}{1 - \exp\left(-\frac{\|\boldsymbol{v}\|^2}{2r^2} \right)} = \frac{\rho_r(t\boldsymbol{v})}{\rho_r(\boldsymbol{v}) - \rho_r(2\boldsymbol{v})}
\end{aligned}$$

as claimed. □

D.2 Proof of Lemma 3

Here, we prove Lemma 3. For reference, we begin by restating the lemma.

Lemma 11. *Let $\mathcal{K}_{\boldsymbol{v}}$ be the set of $\boldsymbol{y}$ such that $|\langle \boldsymbol{y}, \boldsymbol{v}\rangle| \leq \|\boldsymbol{v}\|^2$ and let $\alpha = r/\|\boldsymbol{v}\|$. Then*

$$\max_{\boldsymbol{y}\in\mathcal{K}_{\boldsymbol{v}}} 1 - \sum_{k\in\mathbb{Z}} (-1)^k \frac{\rho_r(\boldsymbol{y}+k\boldsymbol{v})}{\rho_r(\boldsymbol{y})} \leq 1 + \frac{2\alpha\sqrt{2\pi}\rho(\pi\alpha)}{\rho_\alpha(1)\cdot(1-\rho_1(2\pi\alpha))} = M_\alpha$$

and M_α is strictly decreasing with α.

Proof. To begin with, we observe that

$$\begin{aligned}\sum_{k\in\mathbb{Z}} (-1)^k \frac{\rho_r(\boldsymbol{y}+k\boldsymbol{v})}{\rho_r(\boldsymbol{y})} &= \sum_{k\in\mathbb{Z}} \frac{\rho_r(\boldsymbol{y}+2k\boldsymbol{v}) - \rho_r(\boldsymbol{y}-(2k+1)\boldsymbol{v})}{\rho_r(\boldsymbol{y})} \\ &= 2F(2\mathbb{Z}) - F(\mathbb{Z})\end{aligned}$$

where we define

$$\begin{aligned}F(j) &= \frac{\rho_r(\boldsymbol{y}+j\boldsymbol{v})}{\rho_r(\boldsymbol{y})} \\ &= \exp\left(-\frac{j^2\|\boldsymbol{v}\|^2 + 2j\langle \boldsymbol{y},\boldsymbol{v}\rangle}{2r^2}\right) \\ &= \frac{\rho_{r/\|\boldsymbol{v}\|}(j + \langle \boldsymbol{y},\boldsymbol{v}\rangle/\|\boldsymbol{v}\|^2)}{\rho_{r\|\boldsymbol{v}\|}(\langle \boldsymbol{y},\boldsymbol{v}\rangle)}.\end{aligned}$$

With this definition of $F(j)$, we thus want to bound $1 + F(\mathbb{Z}) - 2F(2\mathbb{Z})$. By the Poisson summation formula we have that

$$F(\mathbb{Z}) = \hat{F}(\mathbb{Z}) \text{ and } 2F(2\mathbb{Z}) = \hat{F}(\mathbb{Z}/2)$$

where $\hat{F}$ is the Fourier transform of F. As

$$f(\boldsymbol{y}) = \rho_r(\boldsymbol{y}) = \exp(-\|\boldsymbol{y}\|^2/(2r^2))$$

has the Fourier transform $\hat{f}(\boldsymbol{w}) = r\sqrt{2\pi}\rho_{1/r}(2\pi\boldsymbol{w})$, we have that

$$\hat{F}(k) = \frac{\sqrt{2\pi}\cdot r}{\|\boldsymbol{v}\|\rho_{r\|\boldsymbol{v}\|}\left(\langle \boldsymbol{y},\boldsymbol{v}\rangle\right)} \cdot \exp\left(2\pi i \frac{k\langle \boldsymbol{y},\boldsymbol{v}\rangle}{\|\boldsymbol{v}\|^2}\right) \cdot \rho_{\|\boldsymbol{v}\|/r}\left(2\pi k\right)$$

This gives that

$$\begin{aligned}\hat{F}(\mathbb{Z}) &= \frac{\sqrt{2\pi}\cdot r}{\|\boldsymbol{v}\|\rho_{r\|\boldsymbol{v}\|}\left(\langle \boldsymbol{y},\boldsymbol{v}\rangle\right)} \sum_{k\in\mathbb{Z}} \exp\left(2\pi i \frac{k\langle \boldsymbol{y},\boldsymbol{v}\rangle}{\|\boldsymbol{v}\|^2}\right) \cdot \rho_{\|\boldsymbol{v}\|/r}\left(2\pi k\right) \\ &= \frac{\sqrt{2\pi}\cdot r}{\|\boldsymbol{v}\|\rho_{r\|\boldsymbol{v}\|}\left(\langle \boldsymbol{y},\boldsymbol{v}\rangle\right)} \left(1 + \sum_{k=1}^{\infty} 2\cos\left(2\pi k \frac{\langle \boldsymbol{y},\boldsymbol{v}\rangle}{\|\boldsymbol{v}\|^2}\right) \cdot \rho_{\|\boldsymbol{v}\|/r}\left(2\pi k\right)\right)\end{aligned}$$

while

$$2F(2\mathbb{Z}) = \frac{\sqrt{2\pi}\cdot r}{\|\boldsymbol{v}\|\rho_{r\|\boldsymbol{v}\|}\left(\langle \boldsymbol{y},\boldsymbol{v}\rangle\right)} \left(1 + \sum_{k=1}^{\infty} 2\cos\left(\pi k \frac{\langle \boldsymbol{y},\boldsymbol{v}\rangle}{\|\boldsymbol{v}\|^2}\right) \cdot \rho_{\|\boldsymbol{v}\|/r}\left(\pi k\right)\right).$$

As such, $F(\mathbb{Z}) - 2F(2\mathbb{Z})$ equals

$$-\frac{\sqrt{2\pi}\cdot r}{\|\boldsymbol{v}\|\rho_{r\|\boldsymbol{v}\|}\left(\langle \boldsymbol{y},\boldsymbol{v}\rangle\right)}\sum_{k=1}^{\infty}2\cos\left((2k-1)\pi\frac{\langle \boldsymbol{y},\boldsymbol{v}\rangle}{\|\boldsymbol{v}\|^2}\right)\cdot\rho_{\|\boldsymbol{v}\|/r}\left((2k-1)\pi\right)$$

which in the relevant regime is maximized when $|\langle \boldsymbol{y},\boldsymbol{v}\rangle| = \|\boldsymbol{v}\|^2$. Thus, it follows that

$$1-\sum_{k\in\mathbb{Z}}(-1)^k\frac{p(\boldsymbol{y}+k\boldsymbol{v})}{p(\boldsymbol{y})} \le 1+\frac{2\sqrt{2\pi}\cdot r}{\|\boldsymbol{v}\|\rho_r(\|\boldsymbol{v}\|)}\sum_{k=1}^{\infty}\rho_{\|\boldsymbol{v}\|/r}\left((2k-1)\pi\right). \quad (11)$$

Finally, introducing $\alpha = r/\|\boldsymbol{v}\|$ gives that

$$\begin{aligned}
1-\sum_{k\in\mathbb{Z}}(-1)^k\frac{p(\boldsymbol{y}+k\boldsymbol{v})}{p(\boldsymbol{y})} &\le 1+\frac{2\sqrt{2\pi}\cdot\alpha}{\rho_\alpha(1)}\sum_{k=1}^{\infty}\rho_{1/\alpha}((2k-1)\pi)\\
&= 1+\frac{2\sqrt{2\pi}\cdot\alpha}{\rho_\alpha(1)}\sum_{k=1}^{\infty}\exp(-\pi^2\alpha^2(4k^2-4k+1)/2)\\
&= 1+\frac{2\sqrt{2\pi}\cdot\alpha\rho(\pi\alpha)}{\rho_\alpha(1)}\sum_{k=1}^{\infty}\exp(-2\pi^2\alpha^2(k^2-k))\\
&\le 1+\frac{2\sqrt{2\pi}\cdot\alpha\rho(\pi\alpha)}{\rho_\alpha(1)}\sum_{k=0}^{\infty}\exp(-2\pi^2\alpha^2 k)\\
&= 1+\frac{2\sqrt{2\pi}\cdot\alpha\rho(\pi\alpha)}{\rho_\alpha(1)}\frac{1}{1-\exp(-2\pi^2\alpha^2)}\\
&= 1+\frac{2\alpha\sqrt{2\pi}\rho(\pi\alpha)}{\rho_\alpha(1)\cdot(1-\rho_1(2\pi\alpha))} = M_\alpha.
\end{aligned}$$

which gives our expression for M_α
To see that M_α is strictly decreasing with α, we note that all factors in the second term, besides α, are decreasing with α. Furthermore, we note that $\alpha/\rho_\alpha(1) = \alpha\exp(1/(2\alpha^2))$ is decreasing with α when $\alpha \le 1$ while, when $\alpha > 1$ the product

$$\alpha\cdot\rho_1(\pi\alpha) = \alpha\exp(-\pi^2\alpha^2/2)$$

is decreasing with α. As such, the second term is decreasing with α for all $\alpha > 0$, and the whole expression is thus also decreasing with α. □

E Parameters for Lattice Estimator

Normal Parametrs

For 120 bits of security

```
Unforgeability
SISParameters(n=512, q=12289, length_bound=5649, m=1024, norm=2, tag=None)
```

```
Strong unforgeability
SISParameters(n=512, q=12289, length_bound=11298, m=1024, norm=2, tag=None)


NTWE security
LWEParameters(n=511, q=12289, Xs=D(σ=2.60), Xe=D(σ=2.60), m=+Infinity, tag=None)
```

For 180 bits of security

```
Unforgeability
SISParameters(n=512, q=50177, length_bound=2946, m=1536, norm=2, tag=None)


Strong unforgeability
SISParameters(n=512, q=50177, length_bound=5892, m=1536, norm=2, tag=None)


NTWE security
LWEParameters(n=1023, q=50177, Xs=D(σ=1.00), Xe=D(σ=1.00), m=+Infinity, tag=None)
```

For 260 bits of security

```
Unforgeability
SISParameters(n=768, q=50177, length_bound=5300, m=2048, norm=2, tag=None)


Strong unforgeability
SISParameters(n=768, q=50177, length_bound=10600, m=2048, norm=2, tag=None)


NTWE security
LWEParameters(n=1279, q=50177, Xs=D(σ=1.50), Xe=D(σ=1.50), m=+Infinity, tag=None)
```

Parametrs Without Rejection

For 120 bits of security

```
Unforgeability
SISParameters(n=512, q=50177, length_bound=10460, m=1280, norm=2, tag=None)


Strong unforgeability
SISParameters(n=512, q=50177, length_bound=20920, m=1280, norm=2, tag=None)


NTWE security
LWEParameters(n=767, q=50177, Xs=D(σ=0.90), Xe=D(σ=0.90), m=+Infinity, tag=None)
```

For 180 bits of security

```
Unforgeability
SISParameters(n=768, q=50177, length_bound=14254, m=1792, norm=2, tag=None)

Strong unforgeability
SISParameters(n=768, q=50177, length_bound=28508, m=1792, norm=2, tag=None)

NTWE security
LWEParameters(n=1023, q=50177, Xs=D(σ=1.00), Xe=D(σ=1.00), m=+Infinity, tag=None)
```

For 260 bits of security

```
Unforgeability
SISParameters(n=1024, q=50177, length_bound=20849, m=2304, norm=2, tag=None)

Strong unforgeability
SISParameters(n=1024, q=50177, length_bound=41698, m=2304, norm=2, tag=None)

NTWE security
LWEParameters(n=1279, q=50177, Xs=D(σ=1.45), Xe=D(σ=1.45), m=+Infinity, tag=None)
```