HAETAE

Algorithm Specifications and Supporting Documentation

Jung Hee Cheon, Hyeongmin Choe, Julien Devevey, Tim Güneysu, Dongyeon Hong,
Markus Krausz, Georg Land, Junbum Shin, Damien Stehlé, MinJune Yi.

May 31, 2023.

1 Introduction

We introduce HAETAE[1], a new post-quantum digital signature scheme, whose security is based on the hardness of the module versions of the lattice problems LWE and SIS [BGV12, LS15]. The scheme design follows the "Fiat-Shamir with Aborts" paradigm [Lyu09, Lyu12], which relies on rejection sampling: rejection sampling is used to transform a signature trial whose distribution depends on sensitive information, into a signature whose distribution can be publicly simulated. Our scheme is in part inspired from CRYSTALS-Dilithium [DKL+18], a post-quantum "Fiat-Shamir with Aborts" signature scheme which was selected for standardization by the American National Institute of Standards and Technology (NIST). HAETAE differs from Dilithium in two major aspects: (i) we use a bimodal distribution for the rejection sampling, like in the BLISS signature scheme [DDLL13], instead of a "unimodal" distribution like Dilithium, (ii) we sample from and reject to hyperball uniform distributions, instead of discrete hypercube uniform distributions. This last aspect also departs from BLISS, which relies on discrete Gaussian distributions, and follows a suggestion from [DFPS22], which studied rejection sampling in lattice-based signatures following the "Fiat-Shamir with Aborts" paradigm.

1.1 Design rationale

A brief recap on Fiat-Shamir with Aborts. The Fiat-Shamir with Aborts paradigm was introduced in lattice-based cryptography in [Lyu09, Lyu12]. The verification key is a pair of matrices $(\mathbf{A}, \mathbf{T} = \mathbf{AS} \bmod q)$, where $\mathbf{A}$ is a uniform matrix modulo some integer q and $\mathbf{S}$ is a small-magnitude matrix that makes up the secret key. A signature for a message M is comprised of an integer vector $\mathbf{z}$ of the form $\mathbf{y} + \mathbf{Sc}$, for some random small-magnitude $\mathbf{y}$ and some small-magnitude challenge $\mathbf{c} = H(\mathbf{Ay} \bmod q, M)$. Rejection sampling is then used to ensure that the distribution of the signature becomes independent from the secret key. Finally, the verification algorithm checks that the vector $\mathbf{z}$ is short and that $\mathbf{c} = H(\mathbf{Az} - \mathbf{Tc} \bmod q, M)$.

Improving compactness. As analyzed in [DFPS22], The choice of the distributions to sample from and reject to has a major impact on the signature size. Dilithium relies on discrete uniform distributions in hypercubes, which makes the scheme easier to implement. However, such distributions are far from optimal in terms of resulting signature sizes. We choose a different trade-off: by losing a little on ease of implementation, we obtain more compact signatures.

Uniform distributions in hyperballs. A possibility would be to consider Gaussian distributions, which are superior to uniform distributions in hypercubes, in terms of resulting signature compactness (see, e.g., [DFPS22]). However, this choice has two downsides. First, the rejection step involves the computation of a transcendental function on an input that depends on the secret key. This is cumbersome to implement and sensitive to side-channel attacks [EFGT17]. Second, since the final signature follows a Gaussian distribution there is a nonzero probability that the final signature is too large and does not pass the verification. The signer must realise that and reject the signature, making the expected number of rejects slightly grow in practice. Uniform distributions over hyperballs have been put forward in [DFPS22] as an alternative choice of distributions leading to signatures with compactness between those obtained with Gaussians and those obtained with hypercube uniforms. Compared to Gaussians, they do not suffer from the afore-mentioned downsides: the rejection step is simply checking whether Euclidean norms are sufficiently small; and as there is no tail, there is no need for an extra rejection step to ensure that verification will pass. HAETAE showcases that this provides an interesting simplicity/compactness compromise.

Bimodal distributions. A modification of Lyubashevsky's signatures [Lyu09, Lyu12] was introduced in [DDLL13]. It allows for the use of bimodal distributions in the signature generation. The signature is now of the form $\mathbf{y}+(-1)^b\mathbf{Sc}$, where $\mathbf{y}$ is sampled from a fixed distribution and $b \in \{0, 1\}$ is sampled uniformly. The

[1] The haetae is a mythical Korean lion-like creature with the innate ability to distinguish right from wrong.

signature is then rejected to a given secret-independent target distribution. To make sure that the verification test passes, computations are performed modulo $2q$ and key generation forces the equality $\mathbf{AS} = q\mathbf{Id}$. It turns out that this modification can lead to more compact signatures than the unimodal setup. In [DDLL13], the authors relied on discrete Gaussian distributions. We instead use uniform distributions over hyperballs: like for Gaussians, switching from unimodal to bimodal for hyperball-uniforms leads to more compact signatures.

Flexible design by working with modules. The original design for BLISS [DDLL13] relies on Ring-LWE and Ring-SIS, and a variant of the key generation algorithm relied on ratios of polynomials, *à la* NTRU. This setup forces to choose a working polynomial ring for any desired security level. In order to offer more flexibility without losing in terms of implementation efficiency, we choose to rely on module lattices, like Dilithium, with a fixed working polynomial ring $\mathcal{R} = \mathbb{Z}[x]/(x^{256} + 1)$ across all security levels. In our instantiations, we target the NIST PQC security levels 2, 3 and 5. Varying the security and updating the parameters is easily achievable and we provide a security estimator that is able to help one reach a given target security.

A compact verification key. The flexibility provided by modules allows us to reduce the verification key size. Instead of taking the challenge $\mathbf{c}$ as a vector over $\mathcal{R}$, we choose it in $\mathcal{R}$: the main condition on the challenge is that it has high min-entropy, which is already the case for binary vectors over $\mathcal{R}$. As a result, the secret $\mathbf{S}$ can be chosen as a vector over $\mathcal{R}$ rather than a matrix. The key-pair equation $\mathbf{AS} = q\mathbf{Id}$ then becomes $\mathbf{As} = q\mathbf{j}$, where $\mathbf{j}$ is the vector starting with 1 and then continuing with 0's. To further compress the verification key, we use verification key truncation adopted from Dilithium by taking into account the residue modulo 2. Our key generation algorithm just creates an MLWE sample $(\mathbf{A}_{\mathsf{gen}}, \mathbf{b}-\mathbf{a} = \mathbf{A}_{\mathsf{gen}}\mathbf{s}_{\mathsf{gen}}+\mathbf{e}_{\mathsf{gen}})$ modulo q, where $\mathbf{a}$ is uniform random over $\mathcal{R}_q^k$. By truncating $\mathbf{b}$ as $\mathbf{b} = \mathbf{b}_1+\mathbf{b}_0$, we define a $k\times(k+\ell)$ matrix $\mathbf{A}$ as $\mathbf{A} = (-2(\mathbf{a}-\mathbf{b}_1)+q\mathbf{j}|\ 2\mathbf{A}_{\mathsf{gen}}|\ 2\mathbf{Id}_k) \bmod 2q$. The key-pair equation is satisfied for $\mathbf{s} = (1|\ \mathbf{s}_{\mathsf{gen}}|\ \mathbf{e}_{\mathsf{gen}}-\mathbf{b}_0)$. The verification key consists of $(\mathbf{A}_{\mathsf{gen}}, \mathbf{a}, \mathbf{b}_1)$. As $(\mathbf{a}|\ \mathbf{A}_{\mathsf{gen}})$ is uniformly distributed, we can generate it from a seed using an extendable output function, and the verification key is reduced to the seed and the vector $\mathbf{b}_1$. If we had kept the original key-pair equation $\mathbf{AS} = q\mathbf{Id}$, then the appropriately modified variant of our key-generation algorithm would have led to a verification key that is a matrix (with a seed) rather than a vector (with a seed).

Compression techniques to lower the signature size. We use two techniques to compress the signatures. First, as the verification key $\mathbf{A}$ is in (almost)-HNF, we can use the Bai-Galbraith technique [BG14]. Namely, the second part of the signature, which is multiplied by $2\mathbf{Id}$ in the challenge computation and verification algorithm, can be aggressively compressed by cutting its low bits. This requires in turn modifying the computation of the challenge $\mathbf{c}$ and the verification algorithm, in order to account for this precision loss. Usually, this is done by keeping only the high bits of $\mathbf{Ay}$ in the computation of the challenge. However, as we multiply everything by 2, we do not keep the lowest bit of those high bits and keep the (overall) least significant bit instead. As in Dilithium, our decomposition of bits technique is a Euclidean division with a centered remainder, and we choose a representative range for modular integers that starts slightly below zero to further reduce the support of the high bits. The second compression technique, suggested in [ETWY22] in the context of lattice-based hash-and-sign signatures, concerns the choice of the binary representation of the signature. As the largest part of it consists in a vector that is far from being uniform, we can choose some entropic coding to obtain a signature size close to its entropy. In particular, as in [ETWY22], we choose the efficient range Asymmetric Numeral System to encode our signature, as it allows us to encode the whole signature and not lose a fraction of a bit per vector coordinate, like with Huffman coding. We can further apply the two techniques to the hint vector $\mathbf{h}$, which is also a part of the signature, to reduce the signature sizes.

Efficient choice of modulus. We choose the prime q to be a good prime in the sense that the ring operations can be implemented efficiently and that the decomposition of bits algorithms, are correctly operated. For

ring operations, we use the Number Theoretic Transform (NTT) with a fully splitting polynomial ring. The polynomial ring $\mathcal{R}$ fully splits modulo q when the multiplicative group $\mathbb{Z}_q^\times$ has an element of order 512, or equivalently when $q = 1 \bmod 512$. We choose $q = 64513$, which indeed satisfies this property. Interestingly, it fits in 16 bits, which allows dense storing on embedded devices. Furthermore, it is close to the next power of two, which is convenient for the sampling of uniform integers modulo q.

Fixed-point algorithm for hyperball sampling. Unlike uniform Gaussian sampling or uniform hypercube sampling, uniform hyperball sampling has not been considered in the cryptographic protocols before the suggestion of [DFPS22]. To narrow the gap between the hyperball uniforms sampled in the real and the ideal world, we discretize the hyperball and bound the numerical error and their effect by analyzing their propagation. This leads to a fixed-point hyperball sampling algorithm and, therefore, the fixed-point implementation of the whole signing process.

Deterministic and randomized version. HAETAE can be set in a deterministic or randomized mode. We focus on the deterministic version, but we also give the randomized version. Note that in the randomized version, a significant part of the signing algorithm can be executed off-line as it does not depend on the message.

We give estimated security as well as sizes for our parameter sets in Table 1. The full parameters sets can be found in Section 3.5. The security of our signature is stated in terms of Core-SVP hardness, as introduced in [ADPS16]. We target the core-SVP classical hardness of the known attacks against the three proposed instantiations of HAETAE to be at least 120, 180 and 260, respectively. The numbers between parentheses refer to the strong unforgeability in the case of the randomized version of the signature scheme (for the deterministic version, strong and weak unforgeability are the same). The parameter M refers to the number of average rejections during signing. The KeyRate is the key rejection rate in the key generation algorithm. The parameter η refers to the infinity norm of the secret key $\mathbf{s}_{\mathsf{gen}}$. The parameter τ refers to the Hamming weight of the binary challenge $c \in \mathcal{R}$. The parameter d refers to the bit truncated from the verification key. The sizes are given in bytes. For the signature sizes, we give the average signature sizes when using rANS coding.

1.2 Advantages and limitations

Advantages

- Our scheme relies on the difficulty of hard lattice problems, which have been well-studied for a long time.
- Signature sizes are 30% to 40% smaller than those of Dilithium at comparable security levels, and verification keys are 20% to 25% smaller.
- Implementation-wise, while our design rationale departs from Dilithium's, the scheme remains implementation-friendly. In particular,
  - the rejection step only involves computations of Euclidean norms,
  - the whole signing process can be implemented with fixed-point arithmetic
  - a significant message-independent part of signing can be performed "off-line", for the randomized version of the scheme.

Comparison with hash-and-sign lattice signatures. In terms of ease of implementation, our scheme favorably compares to lattice signatures based on the hash and sign paradigm such as Falcon [FHK+17] and Mitaka [EFG+22a]. HAETAE, Falcon and Mitaka all three rely on some form of Gaussian sampling, which are typically difficult to implement and protect against side-channel attacks. Falcon makes sequential calls to a Gaussian sampler over $\mathbb{Z}$ with arbitrary centers. Mitaka also relies on an integer Gaussian sampler with arbitrary centers, but the calls to it can be massively parallelized. It also uses a continuous Gaussian sampler, which is arguably simpler. HAETAE, however, only relies on a (zero-centered) continuous Gaussian

Parameter set	HAETAE-120	HAETAE-180	HAETAE-260
NIST Security level	2	3	5
q	64513	64513	64513
M	6.0	5.0	6.0
Key Rate	0.1	0.25	0.1
(k, ℓ)	(2,4)	(3,6)	(4,7)
η	1	1	1
τ	58	80	128
α_{h}	512	512	256
d	1	1	0
Forgery			
BKZ block-size b	409 (333)	617 (512)	878 (735)
Classical hardness	119 (97)	180 (149)	256 (214)
Quantum hardness	105 (85)	158 (131)	225 (188)
Key-Recovery			
BKZ block-size b	428	810	988
Classical hardness	125	236	288
Quantum hardness	109	208	253
Signature size	1463	2337	2908
Public key size	992	1472	2080
Sum	2455	3809	4988
Private key size	1376	2080	2720

Table 1: HAETAE parameters sets. Hardness is measured with the Core-SVP methodology.

sampler, used to sample uniformly in hyperballs. The calls to it can also be massively parallelized. This difference makes HAETAE possible to have a fixed-point signing algorithm and easier maskings. Further, in the randomized version of the signature scheme, these samples can be computed off-line as they are independent from the message to be signed. The on-line tasks are far simpler than those of Falcon and Mitaka. Finally, we note that key-generation is much simpler for HAETAE than in Falcon and Mitaka.

Limitations

- The key generation algorithm restarts if the secret key does not satisfy the key rejection condition. This makes the key generation algorithm of HAETAE slower than Dilithium's.
- While HAETAE is simpler from an implementation perspective, its verification key and signature sizes are larger than Falcon's and Mitaka's.

2 Preliminaries

2.1 Notations

Matrices are denoted in bold font and upper case letters (e.g., $\mathbf{A}$), while vectors are denoted in bold font and lowercase letters (e.g., $\mathbf{y}$ or $\mathbf{z}_1$). The i-th component of a vector is denoted with subscript i (e.g., y_i for the i-th component of $\mathbf{y}$).

Every vector is a column vector. We denote concatenation between vectors by putting the rows below as $(\mathbf{u}, \mathbf{v})$ and the columns on the right as $(\mathbf{u}|\mathbf{v})$. We naturally extend the latter notation to concatenations between matrices and vectors (e.g., $(\mathbf{A}|\mathbf{b})$ or $(\mathbf{A}|\mathbf{B})$).

We let $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$ be a polynomial ring where n is a power of 2 integer and for any positive integer q the quotient ring $\mathcal{R}_q = \mathbb{Z}[x]/(q, x^n + 1) = \mathbb{Z}_q[x]/(x^n + 1)$. We abuse notations and identify $\mathcal{R}_2$ with the set of elements in $\mathcal{R}$ with binary coefficients. We also let $\mathcal{R}_\mathbb{R} = \mathbb{R}[x]/(x^n + 1)$ be a polynomial ring over real numbers. For an integer η, we let S_η denote the set of polynomials of degree less than n with coefficients in $[-\eta, \eta] \cap \mathbb{Z}$. Given $\mathbf{y} = (\sum_{0 \le i < n} y_i \; x^i, \cdots, \sum_{0 \le i < n} y_{nk-n+i} \; x^i)^\top \in \mathcal{R}^k$ (or $\mathcal{R}_\mathbb{R}^k$), we define its ℓ_2-norm as the ℓ_2-norm of the corresponding "flattened" vector $\|\mathbf{y}\|_2 = \|(y_0, \cdots, y_{nk-1})^\top\|_2$.

Let $\mathcal{B}_{\mathcal{R},m}(r, \mathbf{c}) = \{\mathbf{x} \in \mathcal{R}_\mathbb{R}^m | \|\mathbf{x} - \mathbf{c}\|_2 \le r\}$ denote the continuous hyperball with center $\mathbf{c} \in \mathcal{R}^m$ and radius $r > 0$ in dimension $m > 0$. When $\mathbf{c} = \mathbf{0}$, we omit it. Let $\mathcal{B}_{(1/N)\mathcal{R},m}(r, \mathbf{c}) = (1/N)\mathcal{R}^m \cap \mathcal{B}_{\mathcal{R},m}(r, \mathbf{c})$ denote the discretized hyperball with radius $r > 0$ and center $\mathbf{c} \in \mathcal{R}^m$ in dimension $m > 0$ with respect to a positive integer N. When $\mathbf{c} = \mathbf{0}$, we omit it. Given a measurable set $X \subseteq \mathcal{R}^m$ of finite volume, we let $U(X)$ denote the continuous uniform distribution over X. It admits $\mathbf{x} \mapsto \chi_X(\mathbf{x})/\mathsf{Vol}(X)$ as a probability density, where χ_X is the indicator function of X and $\mathsf{Vol}(X)$ is the volume of the set X. For the normal distribution over $\mathbb{R}$ centered at μ with standard deviation σ, we use the notation $\mathcal{N}(\mu, \sigma)$.

For a positive integer α, we define $r \bmod^\pm \alpha$ as the unique integer r' in the range $[-\alpha/2, \alpha/2)$ satisfying the relation $r = r' \bmod \alpha$. We also define $r \bmod^+ \alpha$ as the unique integer r' in the range $[0, \alpha)$ that satisfies $r = r' \bmod \alpha$. We denote the least significant bit of an integer r with $\mathsf{LSB}(r)$. We naturally extend this to integer polynomials and vectors of integer polynomials, by applying it component-wise.

2.2 Lattice assumptions

We first recall the well-known lattice assumptions MLWE and MSIS on algebraic lattices.

Definition 1 (Decision-$\mathsf{MLWE}_{n,q,k,\ell,\eta}$). *For positive integers q, k, ℓ, η and the dimension n of $\mathcal{R}$, we say that the advantage of an adversary $\mathcal{A}$ solving the decision-$\mathsf{MLWE}_{n,q,k,\ell,\eta}$ problem is*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{MLWE}}_{n,q,k,\ell,\eta}(\mathcal{A}) = \big| &\Pr\left[b = 1 \mid \mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}; \mathbf{b} \leftarrow \mathcal{R}_q^k; b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{b})\right] \\ &- \Pr\left[b = 1 \mid \mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}; (\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^\ell \times S_\eta^k; b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)\right] \big|.\end{aligned}$$

Definition 2 (Search-$\mathsf{MSIS}_{n,q,k,\ell,\beta}$). *For positive integers q, k, ℓ, a positive real number β and the dimension n of $\mathcal{R}$, we say that the advantage of an adversary $\mathcal{A}$ solving the search-$\mathsf{MSIS}_{n,q,k,\ell,\beta}$ problem is*

$$\mathsf{Adv}^{\mathsf{MSIS}}_{n,q,k,\ell,\beta}(\mathcal{A}) = \Pr\left[\begin{array}{c} 0 < \|\mathbf{y}\|_2 < \beta \wedge \\ (\mathbf{A} \mid \mathbf{Id}_k) \cdot \mathbf{y} = 0 \mod q \end{array} \middle| \mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}; \mathbf{y} \leftarrow \mathcal{A}(\mathbf{A}) \right].$$

2.3 Bimodal hyperball rejection sampling

Recently, Devevey et al. [DFPS22] conducted a study of rejection sampling in the context of lattice-based Fiat-Shamir with Aborts signatures. They observe that (continuous) uniform distributions over hyperballs can be used to obtain compact signatures, with a relatively simple rejection procedure. To make masking easier, HAETAE uses (discretized) uniform distributions over hyperballs, in the bimodal context. The proof of the following lemma is available in Appendix B.

Lemma 1 (Bimodal Hyperball Rejection Sampling). *Let n be the degree of $\mathcal{R}$, $c > 1$, $r, t, m > 0$, and $r' \geq \sqrt{r^2 + t^2}$. Define $M = 2(r'/r)^{mn}$ and set*

$$N \geq \frac{1}{c^{1/(mn)} - 1} \frac{\sqrt{mn}}{2} \left(\frac{c^{1/(mn)}}{r} + \frac{1}{r'} \right).$$

Let $\mathbf{v} \in \mathcal{R}^m \cap \mathcal{B}_{(1/N)\mathcal{R},m}(t)$. Let $p : \mathbb{R}^m \to \{0, 1/2, 1\}$ be defined as follows

$$p(\mathbf{z}) = \begin{cases} 0 & \text{if } \|\mathbf{z}\| \geq r, \\ 1/2 & \text{else if } \|\mathbf{z} - \mathbf{v}\| < r' \wedge \|\mathbf{z} + \mathbf{v}\| < r', \\ 1 & \text{otherwise.} \end{cases}$$

Then there exists $M' \leq cM$ such that the output distributions of the two algorithms from Figure 2 are identical.

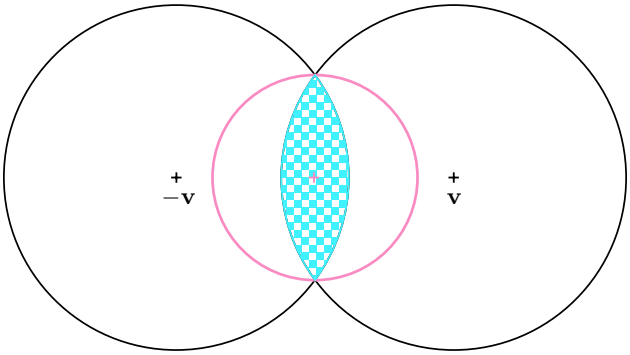


Fig. 1: The HAETAE eyes

Figure 1 illustrates (the continuous version) of the rejection sampling that we consider. The black circles have radii equal to r' and the pink circle has radius r. We sample a vector $\mathbf{z}$ uniformly inside one of the black circles (with probability 1/2 for each) and keep $\mathbf{z}$ with $p(\mathbf{z}) = 1/2$ if $\mathbf{z}$ lies in the blue zone, with probability $p(\mathbf{z}) = 1$ if it lies inside the pink circle but not in the blue zone, and with probability $p(\mathbf{z}) = 0$ everywhere else.

$\mathcal{A}(\mathbf{v})$:	$\mathcal{B}$:
1: $\mathbf{y} \leftarrow U(\mathcal{B}_{(1/N)\mathcal{R},m}(r'))$	1: $\mathbf{z} \leftarrow U(\mathcal{B}_{(1/N)\mathcal{R},m}(r))$
2: $\mathbf{b} \leftarrow U(\{0,1\})$	2: return $\mathbf{z}$ with probability $1/M'$
3: $\mathbf{z} \leftarrow \mathbf{y} + (-1)^b \mathbf{v}$	3: **else** return $\perp$
4: return $\mathbf{z}$ with probability $p(\mathbf{z})$	
5: **else** return $\perp$	

Fig. 2: Bimodal hyperball rejection sampling

2.4 Sampling in a continuous hyperball-uniform

We explain how to sample from a uniform continuous hyperballl distribution. Multiple strategies exist, and the one we choose is such that a k-dimensional module sample is obtained using only $kn+2$ one-dimensional continuous Gaussian samples:

$\mathbf{y} \leftarrow U(\mathcal{B}_{\mathcal{R},k}(r''))$

1: $y_i \leftarrow \mathcal{N}(0,1)$ for $i = 0, \cdots, nk+1$
2: $L \leftarrow \|(y_0, \cdots, y_{nk+1})^\top\|_2$
3: $\mathbf{y} \leftarrow r''/L \cdot (\sum_{i=0}^{n-1} y_i\ x^i, \cdots, \sum_{i=0}^{n-1} y_{nk-n+i}\ x^i)^\top \in \mathcal{R}_{\mathbb{R}}^k$
4: return $\mathbf{y}$

Fig. 3: Continuous hyperball uniform sampling

Lemma 2 ([VGS17]). *The distribution of the output of the algorithm in Figure 3 is $U(\mathcal{B}_{\mathcal{R},k}(r''))$.*

2.5 Challenge sampling

The challenges we use are polynomials $c \in \mathcal{R}$ with binary coefficients and some of them are nonzero. The challenge space has size $\binom{n}{\tau}$ if exactly τ coefficients are nonzero. To sample such challenges we rely on the (binary version of) SampleInBall algorithm from Dilithium, which we recall in Fig. 4.

SampleInBall(ρ, τ)

1: initialize $\mathbf{c} = c_0 c_1 \dots c_{255} = 00 \dots 0$
2: **For** $i = 256 - \tau$ to 255
3: $\quad j \leftarrow \{0, \dots, i\}$
4: $\quad c_i = c_j$
5: $\quad c_j = 1$
6: return $\mathbf{c}$

Fig. 4: Challenge sampling algorithm

For the highest security, however, we require 255 bits of entropy for the challenge, which cannot be reached with $\binom{256}{\tau}$. To achieve it, we replace the challenge sampling for the parameter set with the following. Given a 256-bits hash $w_0 \dots w_{255}$ with Hamming weight w, do the following. If $w < 128$, return $\sum_{i=0}^{255} w_i x^i$. If $w = 128$, return $\sum_{i=0}^{255} w_i \otimes w_0 x^i$. Otherwise, return $\sum_{i=0}^{255} w_i \otimes 1 x^i$. Exactly half of all binary polynomials are reachable this way, which means that the challenge set has size 2^{255} as desired.

2.6 High and low bits

In our scheme, the signature is comprised of a vector $\mathbf{z}$, which we split in two, and a polynomial $\mathbf{c}$. The upper part of $\mathbf{z}$ is split between its high and low bits, and the high bits are compressed. The lower part of $\mathbf{z}$ is not sent, and we instead send a so-called hint. Our technique may be reminiscent of the one from Dilithium [DKL+18], which shares the high-level idea. We first recall the Euclidean division with a centered remainder.

Lemma 3. *Let $a \geq 0$ and $b > 0$. It holds that*

$$a = \left\lfloor \frac{a + b/2}{b} \right\rfloor \cdot b + (a \bmod^{\pm} b),$$

and this writing as $a = bq + r$ with $r \in [-b/2, b/2)$ is unique.

We define our base decomposition function.

Definition 3 (High and low bits). *Let $r \in \mathbb{Z}$ and α be a power of two. Successively define $r_1 = \lfloor (r + \alpha/2)/\alpha \rfloor$ and $r_0 = r \bmod^{\pm} \alpha$. Finally, define the tuple:*

$$(\mathsf{LowBits}(r, \alpha), \mathsf{HighBits}(r, \alpha)) \;=\; (r_0, r_1).$$

We extend these definitions to vectors by applying them component-wise. We state that this decomposition lets us recover the original element and bound the components of the decomposition.

Lemma 4. *Let α be a power of two. Let $q > 2$ be a prime with $\alpha | 2(q-1)$ and $r \in \mathbb{Z}$. Then it holds that*

$$\begin{aligned} r &= \alpha \cdot \mathsf{HighBits}(r, \alpha) + \mathsf{LowBits}(r, \alpha), \\ \mathsf{LowBits}(r, \alpha) &\in [-\alpha/2, \alpha/2), \\ r \in [0, 2q-1] \implies \mathsf{HighBits}(r, \alpha) &\in [0, (2q-1)/\alpha]\,. \end{aligned}$$

Proof. By Lemma 3, there exists a unique representation

$$r = \lfloor (r + \alpha/2)/\alpha \rfloor \, \alpha + (r \bmod^{\pm} \alpha).$$

By identifying $\mathsf{HighBits}(r, \alpha)$ and $\mathsf{LowBits}(r, \alpha)$ in the above equation, we obtain the first result.

Next, by definition of $\bmod^{\pm}$, we have that $r' \in [-\alpha/2, \alpha/2)$.

For the second range, since $r \mapsto \lfloor (r + \alpha/2)/\alpha \rfloor$ is a non-decreasing function, it is sufficient to show that $\lfloor (2q - 1 + \alpha/2)/\alpha \rfloor \leq \lfloor (2q-1)/\alpha \rfloor$. By assumption on q, we have $(2q - 1 + \alpha/2) \leq \lfloor (2q-1)/\alpha \rfloor \alpha + \alpha - 1$. Dividing by α and taking the floor yields the result.

□

We define $\mathsf{HighBits}^{z1}(r) = \mathsf{HighBits}(r, 256)$ and $\mathsf{LowBits}^{z1}(r) = \mathsf{LowBits}(r, 256)$.

High and low bits for hint In order to produce the hint that we send instead of the lower part of $\mathbf{z}$, we could use the previous bit decomposition. However, as noted in [DKL+18, Appendix B] in a preliminary version, a slight modification allows to further reduce the entropy of the hint.

The idea is to pack the high bits in the range $[0, 2(q-1)/\alpha_{\mathsf{h}})$. This is possible if we use the range $[-\alpha_{\mathsf{h}}/2 - 2, 0)$ to represent the integers that are close to $2q - 1$.

Definition 4 (High and low bits for hint). *Let $r \in \mathbb{Z}$. Let q be a prime and $\alpha_{\mathsf{h}} | 2(q-1)$ be a power of two. Let $m = 2(q-1)/\alpha_{\mathsf{h}}$ and*

$$r_1 = \mathsf{HighBits}(r \bmod^{+} 2q, \alpha_{\mathsf{h}}) \quad \textit{and} \quad r_0 = \mathsf{LowBits}(r \bmod^{+} 2q, \alpha_{\mathsf{h}}).$$

If $r_1 = m$, let $(r'_0, r'_1) = (r_0 - 2, 0)$.
Else, $(r'_0, r'_1) = (r_0, r_1)$. We define:

$$(\mathsf{LowBits}^{\mathsf{h}}(r), \mathsf{HighBits}^{\mathsf{h}}(r)) \;=\; (r'_0, r'_1).$$

As before, we extend these definitions to vectors by applying them component-wise. We state that this decomposition lets us recover the original element and bound the decomposition components.

Lemma 5. *Let $r \in \mathbb{Z}$. Let q be a prime, $\alpha_{\mathsf{h}} | 2(q-1)$ be a power of two and define $m = 2(q-1)/\alpha_{\mathsf{h}}$. It holds that*

$$\begin{aligned} r &= \alpha_{\mathsf{h}} \cdot \mathsf{HighBits}^{\mathsf{h}}(r) + \mathsf{LowBits}^{\mathsf{h}}(r) \mod 2q, \\ \mathsf{LowBits}^{\mathsf{h}}(r) &\in [-\alpha_{\mathsf{h}}/2 - 2, \alpha_{\mathsf{h}}/2), \\ \mathsf{HighBits}^{\mathsf{h}}(r) &\in [0, m-1]\,. \end{aligned}$$

Proof. Let $r \in [0, 2q-1]$. Let r_0, r_1, r'_0, and r'_1 defined as in Definition 4. If $r'_0 = r_0$ and $r'_1 = r_1$, the equality $r'_0 + r'_1 \cdot \alpha_{\mathsf{h}} = r_0 + r_1 \cdot \alpha_{\mathsf{h}} \bmod 2q$ holds vacuously.

If not, then $r'_0 = r_0 - 2$ and $r'_1 = r_1 - 2(q-1)/\alpha_{\mathsf{h}}$ and $r'_0 + r'_1\alpha_{\mathsf{h}} = r_0 + r_1\alpha_{\mathsf{h}} - 2q$. By Lemma 4, we get the first equality.

The second property stems from the second property in Lemma 4. The modifications to r_0 make r'_0 lie in the range $[-\alpha_{\mathsf{h}}/2 - 2, \alpha_{\mathsf{h}}/2)$.

The last property stems from the third property in Lemma 4 and the fact that if $r_1 = m$, then we have $r'_1 = 0$.

□

3 Specification

3.1 Key generation

When using bimodal rejection sampling, the verification step relies on a key pair $(\mathbf{A}, \mathbf{s}) \in \mathcal{R}_p^{k\times(k+\ell)} \times \mathcal{R}_p^{k+\ell}$ such that $\mathbf{As} = -\mathbf{As} \bmod p$. To generate such a pair, following [DDLL13], we choose $p = 2q$ and aim at $\mathbf{As} = q\mathbf{j} \bmod 2q$ for $\mathbf{j} = (1, 0, \ldots, 0)^\top$.

Key generation and encoding To build a key pair $(\mathbf{A}, \mathbf{s})$ satisfying the above, we start by generating an MLWE sample $\mathbf{b} - \mathbf{a} = \mathbf{A}_0\mathbf{s}_0 + \mathbf{e}_0 \bmod q$, where $\mathbf{A}_0 \leftarrow U(\mathcal{R}_q^{k\times(\ell-1)})$, $\mathbf{a} \leftarrow U(\mathcal{R}_q^k)$ and $(\mathbf{s}_0, \mathbf{e}_0) \leftarrow U(S_\eta^{\ell-1} \times S_\eta^k)$. For any $\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_0$, we define $\mathbf{A} = (2(\mathbf{a} - \mathbf{b}_1) + q\mathbf{j} | 2\mathbf{A}_0 | 2\mathbf{I}_k)$ as well as $\mathbf{s} = (1 | \mathbf{s}_0 | (\mathbf{e}_0 - \mathbf{b}_0))$. One sees that $\mathbf{As} = q\mathbf{j} \bmod 2q$. In practice, the verification key is then comprised of $\mathbf{b}_1$ and the seed that allows generating $\mathbf{A}_0$ and $\mathbf{a}$. The secret key is the seed used to generate $\mathbf{s}$ and $(\mathbf{A}_0, \mathbf{a})$.

It remains to choose the decomposition of $\mathbf{b}$, that we see as an nk-dimensional vector with coordinates in $[0, q-1]$. We choose $\mathbf{b}_0$ with coordinates in $\{-1, 0, 1\}$ such that if a coordinate of $\mathbf{b}$ is odd, then it is rounded to the nearest multiple of 4. We can then write $\mathbf{b} = \mathbf{b}_0 + 2\mathbf{b}_1$, where $\mathbf{b}_1$ is encoded using $\lceil \log_2(q) - 1 \rceil$ bits per coordinate. This is computed coordinate-wise with $\mathbf{b}_0 = (-1)^{\lfloor \mathbf{b}/2 \rfloor \bmod 2}\mathbf{b} \bmod 2$, i.e. one less bit than $\mathbf{b}$. In all of the following, we let $(\mathsf{LowBits}^{\mathsf{vk}}(\mathbf{b}), \mathsf{HighBits}^{\mathsf{vk}}(\mathbf{b}))$ denote $(\mathbf{b}_0, \mathbf{b}_1)$. When $\mathbf{b}$ is uniform, we notice that the coordinates of $\mathbf{b}_0$ roughly follow a (centered) binomial law with parameters $(2, 1/2)$, which experimentally leads to smaller choices for β, which we discuss and introduce now.

Rejection sampling on the key A critical step of our scheme is bounding $\|\mathbf{s}c\|_2$, where $\mathbf{s}$ is generated as before and $c \in \mathcal{R}$ is a polynomial with coefficients in $\{0, 1\}$ and has less than or equal to τ nonzero coefficients. The lower this bound is, the smaller the signature is, which in turn leads to harder forging. In the key generation algorithm, we apply the following rejection condition for some heuristic value β:

$$\tau \cdot \sum_{i=1}^{m} \max_j{}^{i\text{-th}} \|\mathbf{s}(\omega_j)\|_2^2 + r \cdot \max_j{}^{(m+1)\text{-th}} \|\mathbf{s}(\omega_j)\|_2^2 \leq \frac{n\beta^2}{\tau},$$

where $m = \lfloor n/\tau \rfloor$ and $r = n \bmod \tau$. We argue that the left hand side is a bound on $\frac{n}{\tau} \cdot \|\mathbf{s}c\|_2^2$ and that this condition leads to asserting $\|\mathbf{s}c\|_2 \leq \beta$.

Lemma 6. *For any $c \in \{0,1\}^n$ with hamming weight τ and a secret $\mathbf{s} \in S_\eta^{k+\ell}$, $n\|c\mathbf{s}\|_2^2$ is bounded by*

$$\tau^2 \cdot \sum_{i=1}^{m} \max_j{}^{i\text{-}th} \|\mathbf{s}(\omega_j)\|_2^2 + r \cdot \tau \cdot \max_j{}^{(m+1)\text{-}th} \|\mathbf{s}(\omega_j)\|_2^2,$$

where $m = \lfloor n/\tau \rfloor$ and $r = n \bmod \tau$.

Proof. We first rewrite $\|\mathbf{s}c\|^2$ as:

$$\|\mathbf{s}c\|_2^2 = \frac{\sum_i |c(\omega_j)|^2 \cdot \|\mathbf{s}(\omega_j)\|_2^2}{n},$$

where $\mathbf{s}(\omega_j) = (\mathbf{s}_1(\omega_j), \cdots, \mathbf{s}_{k+\ell}(\omega_j))$, and ω_j's are the primitive $2n$-th roots of unity. Let $m = \lfloor n/\tau \rfloor$ and $r = n \bmod \tau$. Since $\sum_{j=1}^n |c(\omega_j)|^2 = n\tau$ and

$$|c(\omega_j)|^2 = |\ \omega_{j,1} + \cdots + \omega_{j,\tau}|^2 \leq \tau^2,$$

we can bound $\sum_{j=1}^n |c(\omega_j)|^2 \cdot \|\mathbf{s}(\omega_j)\|_2^2$ by rearrangement: let $m = \lfloor n/\tau \rfloor$ be the maximum number of $|c(\omega_j)|^2$'s that can be τ^2. By sorting $\|\mathbf{s}(\omega_j)\|_2$ in a decreasing order,

$$\|\mathbf{s}(\omega_{\sigma(1)})\|_2 \geq \|\mathbf{s}(\omega_{\sigma(2)})\|_2 \geq \cdots \geq \|\mathbf{s}(\omega_{\sigma(n)})\|_2,$$

where σ is a permutation for the indices, we have

$$\sum_{j=1}^{n} |c(\omega_j)|^2 \cdot \|\mathbf{s}(\omega_j)\|_2^2 \leq \sum_{j=1}^{m} |c(\omega_{\sigma(j)})|^2 \cdot \|\mathbf{s}(\omega_{\sigma(j)})\|_2^2 + \sum_{j=m+1}^{n} |c(\omega_{\sigma(j)})|^2 \cdot \|\mathbf{s}(\omega_{\sigma(m+1)})\|_2^2.$$

Then it reaches the maximum when the m largest $\|\mathbf{s}(\omega_j)\|_2^2$'s are multiplied with τ^2's, i.e.,

$$\begin{aligned}\sum_{j=1}^{n} |c(\omega_j)|^2 \cdot \|\mathbf{s}(\omega_j)\|_2^2 &\leq \sum_{j=1}^{m} \tau^2 \cdot \|\mathbf{s}(\omega_{\sigma(j)})\|_2^2 + \Big(\sum_{j=1}^{n} |c(\omega_j)|^2 - m\tau^2\Big) \cdot \|\mathbf{s}(\omega_{\sigma(j)})\|_2^2 \\ &= \tau^2 \cdot \sum_{j=1}^{m} \|\mathbf{s}(\omega_{\sigma(j)})\|_2^2 + r \cdot \tau \cdot \|\mathbf{s}(\omega_{\sigma(j)})\|_2^2.\end{aligned}$$

□

3.2 Discrete hyperball sampling

In orger to generate $\mathbf{y}$ from Figure 2 using the continuous hyperball uniform sampling from Figure 3, we apply a rounding-and-reject strategy which allows to generate rightly distributed samples.

$\underline{\mathbf{y} \leftarrow U(\mathcal{B}_{(1/N)\mathcal{R},m}(r'))}$
1: $\mathbf{y} \leftarrow U(\mathcal{B}_{\mathcal{R},m}(Nr' + \sqrt{mn}/2))$
2: **if** $\|\lfloor \mathbf{y} \rceil\|_2 \leq Nr'$, return $\lfloor \mathbf{y} \rceil / N$
3: **else**, restart

Fig. 5: Discrete hyperball uniform sampling

Lemma 7. *Let n be the degree of $\mathcal{R}$, $M_0 \geq 1$, $r', mi, N > 0$. At each iteration, the algorithm from Figure 5 succeeds with probability $\geq 1/M_0$ and the distribution of the output is $U(\mathcal{B}_{(1/N)\mathcal{R},m}(r'))$ if we set*

$$N \geq \frac{\sqrt{mn}}{2r'} \cdot \frac{M_0^{1/(mn)} + 1}{M_0^{1/(mn)} - 1}.$$

We note that with this rounding step, we do not need to handle the exact values of $\mathbf{y}$, we just need enough precision to make sure the rounding is correct. The proof of this lemma can also be found in Appendix B.

We now have all necessary ingredients in Figures 1, 2, 3, and 5 to make sure the resulting distribution of $\mathbf{z}$ is indeed uniform over the discretized hyperball. Thanks to Lemma 7 and Lemma 1, we already know the level of precision required for $\mathbf{y}$ to maintain the provable security of HAETAE. We analyze in Appendix C the required precision from a fix-point Gaussian sampler to obtain a $\mathbf{y}$ with such precision.

3.3 Signature encoding

To encode a signature, we will split some of its components into low and high bits. If we correctly choose the number of low bits, they will be distributed almost uniformly. The high bits on the other hand, will then follow a distribution with a very small variance and can be considerably compressed with a suitable encoding. While Huffman coding would be applied on each coordinate at a time, an arithmetic coding encodes the entire coordinates in a single number. In contrast to Huffman coding, arithmetic coding gets close to entropy also for alphabets, where the probabilities of the symbols are not powers of two. We recall a recent type

of entropy coding, named range Asymmetric Numeral systems (rANS) [Dud13], that encodes the state in a natural number and thus allows faster implementations. As a stream variant, rANS can be implemented with finite precision integer arithmetic by using renormalization. Furthermore, it is possible to avoid arithmetic operations altogether and realize high-speed implementations using lookup tables (tANS).

Definition 5 (Range Asymmetric Numeral System (rANS) Coding). *Let $n > 0$ and $S \subseteq [0, 2^n - 1]$. Let $g : [0, 2^n - 1] \to \mathbb{Z} \cap (0, 2^n]$ such that $\sum_{x \in S} g(x) \leq 2^n$ and $g(x) = 0$ for all $x \notin S$. We define the following:*

- $\mathsf{CDF} : S \to \mathbb{Z}$, *defined as* $\mathsf{CDF}(s) = \sum_{y=0}^{s-1} g(y)$.
- $\mathsf{symbol} : \mathbb{Z} \to S$, *where* $\mathsf{symbol}(y)$ *is defined as* $s \in S$ *satisfying* $\mathsf{CDF}(s) \leq y < \mathsf{CDF}(s+1)$.
- $C : \mathbb{Z} \times S \to \mathbb{Z}$, *defined as*

$$C(x, s) = \left\lfloor \frac{x}{g(s)} \right\rfloor \cdot 2^n + (x \bmod^+ g(s)) + \mathsf{CDF}(s).$$

Then, we define the rANS encoding/decoding for the set S and frequency $g/2^n$ as in Figure 6.

$\underline{\mathsf{Encode}((s_1, \cdots, s_m) \in S^m)}$	$\underline{\mathsf{Decode}(x \in \mathbb{Z})}$
1: $x_0 = 0$	1: $y_0 = x$
2: **for** $i = 0, \cdots, m-1$ **do**	2: $i = 0$
3: $\quad x_{i+1} = C(x_i, s_{i+1})$	3: **while** $y_i > 0$ **do**
4: Return x_m	4: $\quad t_{i+1} = \mathsf{symbol}(y_i \bmod^+ 2^n)$
	5: $\quad y_{i+1} = \lfloor y_i / 2^n \rfloor \cdot g(t_{i+1}) + (y_i \bmod^+ 2^n) - \mathsf{CDF}(t_{i+1})$
	6: $\quad i \leftarrow i + 1$
	7: $m = i - 1$
	8: return $(t_m, \cdots, t_1) \in S^m$

Fig. 6: rANS encoding and decoding procedures

Lemma 8 (Adapted from [Dud13]). *The rANS coding is correct, and the size of the rANS code is asymptotically equal to Shannon entropy of the symbols. That is, for any choice of $\mathbf{s} = (s_1, \cdots, s_m) \in S^m$, $\mathsf{Decode}(\mathsf{Encode}(\mathbf{s})) = \mathbf{s}$. Moreover, for any positive x and any probability distribution p over S, it holds that*

$$\sum_{s \in S} p(s) \log(C(x, s)) \leq \log(x) + \sum_{s \in S} p(s) \log\left(\frac{g(s)}{2^n}\right) + \frac{2^n}{x}.$$

Finally, the cost of encoding the first symbol is $\leq n$, i.e., for any $x \in S$, we have $\log(C(0, s)) \leq n$.

We determine the frequency of the symbols experimentally, by executing the signature computation and collecting several million samples. Finally, we apply some rounding strategy in order to heuristically minimize the empirical entropy $\sum_{s \in S} p(s) \log(g(s)/2^n)$.

3.4 Specification of HAETAE

Readers who are not familiar with the Fiat-Shamir with Aborts line of work may first check the uncompressed version of the scheme in Appendix A. We give the description of the signature scheme HAETAE in Figure 7 with the following building blocks:

- Hash function H_{gen} for generating the seeds and hashing the messages,
- Hash function H for signing, returning ρ, a seed for challenge sampling,
- Extendable output function expandA for deriving $\mathbf{a}$ and $\mathbf{A}_{\mathsf{gen}}$ from $\mathsf{seed}_{\mathbf{A}}$,

- Extendable output function $\mathsf{expandS}$ for deriving $\mathbf{s}_{\mathsf{gen}}$ and $\mathbf{e}_{\mathsf{gen}}$ from $\mathsf{seed}_{\mathsf{sk}}$ and $\mathsf{counter}_{\mathsf{sk}}$,
- Extendable output function $\mathsf{expandYbb}$ for deriving $\mathbf{y}$, b and b' from seed_{ybb} and $\mathsf{counter}$,

The above building blocks can be implemented with symmetric primitives. Specifically, we use $\mathsf{SHAKE256}$ for the hash functions and Extendable output functions, except for $\mathsf{expandA}$. In all of the following sections, we let $\mathbf{j} = (1, 0, \ldots, 0) \in \mathcal{R}^k$. The parameters ρ_0 and α_{h} refer to the size of the seed and the compression factor, respectively. The parameter β is the bound for $\|c\mathbf{s}\|$, which will be checked by bounding

$$f(\mathbf{s}) \coloneqq \tau \cdot \sum_{i=1}^{m} \max_{j}^{i\text{-th}} \|\mathbf{s}(\omega_j)\|_2^2 + r \cdot \max_{j}^{(m+1)\text{-th}} \|\mathbf{s}(\omega_j)\|_2^2$$

by $n\beta^2/\tau$. The parameters B, B', and B'' refer to the radii of hyperballs. At Step 2 of the Sign algorithm, the variable $y_0 \in \mathcal{R}_{\mathbb{R}}$ refers to the first component of the vector $\mathbf{y} \in \mathcal{R}_{\mathbb{R}}^{k+\ell}$. At Step 3 of the Sign algorithm, the vector $\mathbf{z} \in \mathcal{R}_{\mathbb{R}}^{k+\ell}$ is decomposed as $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2)$ with $\mathbf{z}_1 \in \mathcal{R}_{\mathbb{R}}^{\ell}$ and $\mathbf{z}_2 \in \mathcal{R}_{\mathbb{R}}^{k}$. At Step 4 of the Verify algorithm, the variable $\tilde{z}_0 \in \mathcal{R}$ refers to the first component of the vector $\tilde{\mathbf{z}} \in \mathcal{R}^{k+\ell}$. We assume that q and α_{h} satisfy the assumptions from Lemma 5.

Note that at Step 6 of the Verify algorithm, the division by 2 is well-defined as the operand is even.

We also give a randomized signing of HAETAE in Figure 8. We observe that in the randomized version signing process, significant part of signing including the hyperball sampling algorithms for $\mathbf{y}$ can be performed "off-line", i.e., before receiving a message M to be signed. It holds for computations such as $\mathbf{w} = \mathbf{A}\lfloor \mathbf{y} \rceil$ and $\mathsf{HighBits}^{\mathsf{h}}(\mathbf{w})$. In the "on-line" phase of signing, we can use $\mathbf{y}$ and the corresponging pre-computed components by choosing them randomly among the pre-sampled list.

Lemma 9. *We borrow the notations from Figure 7. If we run* $\mathsf{Verify}(\mathsf{vk}, M, \sigma)$ *on the signature* σ *returned by* $\mathsf{Sign}(\mathsf{sk}, M)$ *for an arbitrary message* M *and an arbitrary key-pair* $(\mathsf{sk}, \mathsf{vk})$ *returned by* $\mathsf{KeyGen}(1^\lambda)$*, then the following relations hold:*

1) $\mathbf{w}_1 = \mathsf{HighBits}^{\mathsf{h}}(\mathbf{w})$,
2) $w'\mathbf{j} = \mathsf{LSB}(\lfloor y_0 \rceil) \cdot \mathbf{j} = \mathsf{LSB}(\mathbf{w}) = \mathsf{LSB}(\mathbf{w} - 2\lfloor \mathbf{z}_2 \rceil)$.
3) $2\lfloor \mathbf{z}_2 \rceil - 2\tilde{\mathbf{z}}_2 = \mathsf{LowBits}^{\mathsf{h}}(\mathbf{w}) - \mathsf{LSB}(\mathbf{w})$ *assuming it holds that* $B' + \alpha_{\mathsf{h}}/4 + 1 \leq B'' < q/2$,

Proof. Let $m = 2(q-1)/\alpha_{\mathsf{h}}$. Let us prove the first statement. By definition of $\mathbf{h}$, it holds that $\mathbf{w}_1 = \mathsf{HighBits}^{\mathsf{h}}(\mathbf{w}) \mod m$. However, the latter part of the equality already lies in $[0, m-1]$ by Lemma 5. The first part lies in the same range as we reduce $\mod^+ m$. Hence, the equality stands over $\mathbb{Z}$ too.

We move on to the second statement. By considering only the first component of $\mathbf{z} = \mathbf{y} + (-1)^b c \cdot \mathbf{s}$, we obtain, modulo 2:

$$\tilde{z}_0 = \lfloor z_0 \rceil = \lfloor y_0 \rceil + (-1)^b c = \lfloor y_0 \rceil + c.$$

This yields the result. Moreover, considering everywhere a 2 appears in the definition of $\mathbf{A}$, we obtain that

$$\mathbf{w} = \mathbf{A}_1 \lfloor \mathbf{z}_1 \rceil - qc\mathbf{j} = (\lfloor z_0 \rceil - c)\mathbf{j} \bmod 2.$$

For the last statement, let us use the two preceding results. In particular, we note the identity

$$\mathbf{w}_1 \cdot \alpha_{\mathsf{h}} + w'\mathbf{j} = \mathbf{w} - \mathsf{LowBits}^{\mathsf{h}}(\mathbf{w}) + \mathsf{LSB}(\mathbf{w}).$$

We note that the last two elements have same parity, as the former one has the same parity as $\mathsf{LowBits}(\mathbf{w}, \alpha_{\mathsf{h}})$. By Lemma 5 their sum has infinite norm $\leq \alpha_{\mathsf{h}}/2 + 2$. Hence from its definition, it holds that

$$2\tilde{\mathbf{z}}_2 = 2\lfloor \mathbf{z}_2 \rceil - \mathsf{LowBits}^{\mathsf{h}}(\mathbf{w}) + \mathsf{LSB}(\mathbf{w}) \bmod^{\pm} 2q.$$

Finally, this holds over the integers as the right-hand side has infinite norm at most $2B' + \alpha_{\mathsf{h}}/2 + 2 < q$. □

Theorem 1 (Completeness). *Assume that* $B'' = B' + \sqrt{n(k+\ell)}/2 + \sqrt{nk} \cdot (\alpha_{\mathsf{h}}/4 + 1) < q/2$*. Then the signature schemes of Figures 7 and 8 are complete, i.e., for every message* M *and every key-pair* $(\mathsf{sk}, \mathsf{vk})$ *returned by* $\mathsf{KeyGen}(1^\lambda)$*, we have* $\mathsf{Verify}(\mathsf{vk}, M, \mathsf{Sign}(\mathsf{sk}, M)) = 1$.

$\underline{\mathsf{KeyGen}(1^\lambda)}$

1: $\mathsf{seed} \leftarrow \{0,1\}^{\rho_0}$
2: $(\mathsf{seed}_{\mathbf{A}}, \mathsf{seed}_{\mathsf{sk}}, K) = H_{\mathsf{gen}}(\mathsf{seed})$
3: $(\mathbf{a}|\ \mathbf{A}_{\mathsf{gen}}) \in \mathcal{R}_q^{k\times\ell} \coloneqq \mathsf{expandA}(\mathsf{seed}_{\mathbf{A}})$
4: $\mathsf{counter}_{\mathsf{sk}} = 0$
5: $(\mathbf{s}_{\mathsf{gen}}, \mathbf{e}_{\mathsf{gen}}) \coloneqq \mathsf{expandS}(\mathsf{seed}_{\mathsf{sk}}, \mathsf{counter}_{\mathsf{sk}})$
6: $\mathbf{b} = \mathbf{a} + \mathbf{A}_{\mathsf{gen}} \cdot \mathbf{s}_{\mathsf{gen}} + \mathbf{e}_{\mathsf{gen}} \bmod q$ // $\mathbf{b} \in \mathcal{R}_q^k$
7: $(\mathbf{b}_0, \mathbf{b}_1) = (\mathsf{LowBits}^{\mathsf{vk}}(\mathbf{b}), \mathsf{HighBits}^{\mathsf{vk}}(\mathbf{b}))$
8: $\mathbf{A} = (2(\mathbf{a} - \mathbf{b}_1) + q\mathbf{j}|\ 2\mathbf{A}_{\mathsf{gen}}|\ 2\mathbf{Id}_k) \bmod 2q$ // $\mathbf{A} \in \mathcal{R}_{2q}^{k\times(k+\ell)}$
9: $\mathbf{s} = (1, \mathbf{s}_{\mathsf{gen}}, \mathbf{e}_{\mathsf{gen}} - \mathbf{b}_0)$ // $\mathbf{s} \in S_\eta^{k+\ell}$
10: **if** $f(\mathbf{s}) > n\beta^2/\tau$ then go to 5
11: return $\mathsf{sk} = (\mathbf{s}, K)$, $\mathsf{vk} = (\mathsf{seed}_{\mathbf{A}}, \mathbf{b}_1)$

$\underline{\mathsf{Sign}(\mathsf{sk}, M)}$

1: $\mu = H_{\mathsf{gen}}(\mathsf{seed}_{\mathbf{A}}, \mathbf{b}_1, M)$
2: $\mathsf{seed}_{ybb} = H_{\mathsf{gen}}(K, \mu)$
3: $\mathsf{counter} = 0$
4: $(\mathbf{y}, b, b') \coloneqq \mathsf{expandYbb}(\mathsf{seed}_{ybb}, \mathsf{counter})$
5: $\mathbf{w} = \mathbf{A}\ \lfloor \mathbf{y} \rceil$
6: $\rho = H(\mathsf{HighBits}^{\mathsf{h}}(\mathbf{w}), \mathsf{LSB}(\lfloor y_0 \rceil), \mu)$
7: $c = \mathsf{SampleInBall}(\rho, \tau)$
8: $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2) = \mathbf{y} + (-1)^b c \cdot \mathbf{s}$
9: $\mathbf{h} = \mathsf{HighBits}^{\mathsf{h}}(\mathbf{w}) - \mathsf{HighBits}^{\mathsf{h}}(\mathbf{w} - 2\ \lfloor \mathbf{z}_2 \rceil) \bmod^+ \frac{2(q-1)}{\alpha_{\mathsf{h}}}$
10: **if** $\|\mathbf{z}\|_2 \geq B'$, then $\mathsf{counter}{+}{+}$ and go to 4
11: **else if** $\|2\mathbf{z} - \mathbf{y}\|_2 < B$ **and** $b' = 0$, then $\mathsf{counter}{+}{+}$ and go to 4
12: **else** return $\sigma = (\mathsf{Encode}(\mathsf{HighBits}^{z1}(\lfloor \mathbf{z}_1 \rceil)), \mathsf{LowBits}^{z1}(\lfloor \mathbf{z}_1 \rceil), \mathsf{Encode}(\mathbf{h}), c)$

$\underline{\mathsf{Verify}(\mathsf{vk}, M, \sigma = (x, \mathbf{v}, h, c))}$

1: $\tilde{\mathbf{z}}_1 \leftarrow \mathsf{Decode}(x) \cdot a + \mathbf{v}$ and $\tilde{\mathbf{h}} = \mathsf{Decode}(h)$
2: $(\mathbf{a}|\ \mathbf{A}_{\mathsf{gen}}) = \mathsf{expandA}(\mathsf{seed}_{\mathbf{A}})$
3: $\mathbf{A}_1 = (2(\mathbf{a} - 2\mathbf{b}_1) + q\mathbf{j}|\ 2\mathbf{A}_{\mathsf{gen}})$
4: $\mathbf{w}_1 = \tilde{\mathbf{h}} + \mathsf{HighBits}^{\mathsf{h}}(\mathbf{A}_1\tilde{\mathbf{z}}_1 - qc\mathbf{j}) \bmod^+ \frac{2(q-1)}{\alpha_{\mathsf{h}}}$
5: $w' = \mathsf{LSB}(\tilde{z}_0 - c)$
6: $\tilde{\mathbf{z}}_2 = [\mathbf{w}_1 \cdot \alpha_{\mathsf{h}} + w'\mathbf{j} - (\mathbf{A}_1\tilde{\mathbf{z}}_1 - qc\mathbf{j})]/2 \bmod^{\pm} q$
7: $\tilde{\mathbf{z}} = (\tilde{\mathbf{z}}_1, \tilde{\mathbf{z}}_2)$
8: $\tilde{\mu} = H_{\mathsf{gen}}(\mathsf{seed}_{\mathbf{A}}, \mathbf{b}_1, M)$
9: Return $(c = \mathsf{SampleInBall}(H(\mathbf{w}_1, w', \tilde{\mu}), \tau)) \wedge (\|\tilde{\mathbf{z}}\| < B'')$

Fig. 7: Deterministic version of HAETAE

Proof. We use the notations of the algorithms. We will focus on the deterministic version in Fig. 7, since Fig. 8 also has almost the same proof. The first and second equations from Lemma 9 state that $\rho = \tilde{\rho}$ and thus $c = \mathsf{SampleInBall}(\rho, \tau)$.

On the other hand, we use the last equation from the same lemma to bound the size of $\tilde{\mathbf{z}}$. We have:

$$\begin{aligned}
\|\tilde{\mathbf{z}}\| &\leq \|\mathbf{z}\| + \|\mathbf{z} - \lfloor \mathbf{z} \rceil\| + \|\lfloor \mathbf{z} \rceil - \tilde{\mathbf{z}}\| \\
&\leq B' + \sqrt{n(k+\ell)} \cdot \|\mathbf{z} - \lfloor \mathbf{z} \rceil\|_\infty + \|\lfloor \mathbf{z}_2 \rceil - \tilde{\mathbf{z}}_2\| \\
&\leq B' + \frac{\sqrt{n(k+\ell)}}{2} + \sqrt{nk} \cdot \|\mathsf{LowBits}^{\mathsf{h}}(\mathbf{w})\|_\infty \\
&\leq B' + \frac{\sqrt{n(k+\ell)}}{2} + \sqrt{nk} \cdot \left(\frac{\alpha_{\mathsf{h}}}{4} + 1\right).
\end{aligned}$$

The definition of B'' implies that the scheme is correct. □

$\underline{\mathsf{Sign}(\mathsf{sk}, M)}$

// can be done off-line: using vk, make a list $\mathcal{L}$ of $(\mathbf{y}, \mathbf{w}, \mathbf{w}_1)$

1: $\mathbf{y} \leftarrow U(\mathcal{B}_{(1/N)\mathcal{R},(k+\ell)}(B))$
2: $\mathbf{w} = \mathbf{A}\lfloor \mathbf{y} \rceil$
3: $\mathbf{w}_1 = \mathsf{HighBits}^{\mathsf{h}}(\mathbf{w})$

// can be done on-line: using sk, M and pre-computed $(\mathbf{y}, \mathbf{w}, \mathbf{w}_1)$ sampled
// from $\mathcal{L}$

4: $\mu = H_{\mathsf{gen}}(\mathsf{seed}_{\mathbf{A}}, \mathbf{b}_1, M)$
5: $b, b' \leftarrow \{0, 1\}$
6: $c = \mathsf{SampleInBall}(H(\mathbf{w}_1, \mathsf{LSB}(\lfloor y_0 \rceil), \mu), \tau)$
7: $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2) = \mathbf{y} + (-1)^b c \cdot \mathbf{s}$
8: $\mathbf{h} = \mathbf{w}_1 - \mathsf{HighBits}^{\mathsf{h}}(\mathbf{w} - 2\lfloor \mathbf{z}_2 \rceil) \bmod^{+} \frac{2(q-1)}{\alpha_{\mathsf{h}}}$
9: **if** $\|\mathbf{z}\|_2 \geq B'$, then
10: go to 5 with resampled $(\mathbf{y}, \mathbf{w}, \mathbf{w}_1)$ // resample $(\mathbf{y}, \mathbf{w}, \mathbf{w}_1) \leftarrow \mathcal{L}$
11: **else if** $(\|2\mathbf{z} - \mathbf{y}\|_2 < B) \wedge (b' = 0)$, then
12: go to 5 with resampled $(\mathbf{y}, \mathbf{w}, \mathbf{w}_1)$ // resample $(\mathbf{y}, \mathbf{w}, \mathbf{w}_1) \leftarrow \mathcal{L}$
13: **else** return $\sigma = (\mathsf{Encode}(\mathsf{HighBits}^{z1}(\lfloor \mathbf{z}_1 \rceil)), \mathsf{LowBits}^{z1}(\lfloor \mathbf{z}_1 \rceil), \mathsf{Encode}(\mathbf{h}), c)$

Fig. 8: Randomized signing of HAETAE. On/offline signing can accelerate the signing process. Note that the signing can also be accelerated even if $\mathbf{y}$ is sampled offline alone.

3.5 Parameter sets and signature sizes

We propose three different parameter sets with varying security levels, where we prioritize low signature and verification key sizes over faster execution time. The parameter choices are versatile, adaptable and allow size vs. speed trade-offs at consistent security levels. For example at cost of larger signatures, a smaller repetition rate M is possible and thus a faster signing process. This versatility is a notable advantage over FALCON and MITAKA.

Like in DILITHIUM, our modulus q is constant over the parameter sets and allows an optimized NTT implementation shared for all sets. With only 16-bit in size, our modulus also allows storing coefficients memory-efficiently without compression.

The rANS encoded values h and high bits of $\mathbf{z}_1$ lead to a varying signature size. In our current implementation we opted for a fixed signature size as reported in Table 3. We evaluated the distribution empirically and determined a threshold that requires a rejection in less than 0.1% of the cases. A field of two bytes indicates the length of the encoded values, the padding can be done with arbitrary data.

A dynamic signature size would allow an individual implementation to reject and recompute signatures until a desired size threshold is reached and still be compatible with implementations without this rejection. Due to the small variance in the distribution of the signature size, however, this would result in a distinct performance overhead, if the threshold is more than a few bytes below to the average size. Figure 9 displays the signature size distribution of 1000 executions.

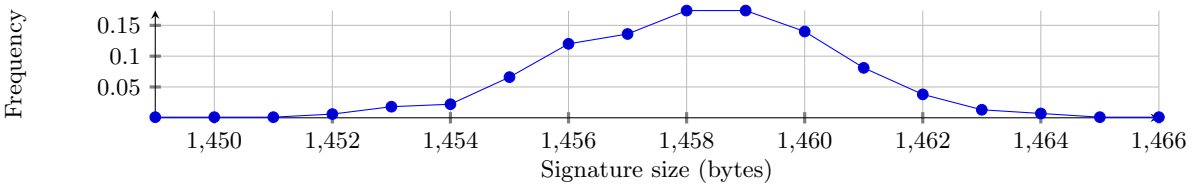
In Table 3 we compare the signature and key sizes of HAETAE, DILITHIUM, and FALCON. The verification keys in HAETAE are 20% (HAETAE-260) to 25% (HAETAE-120 and HAETAE-180) smaller, than their counterparts in DILITHIUM. The advantage of the hyperball sampling manifests itself in the signature sizes, HAETAE has 30% to 40% smaller signatures than DILITHIUM. Less relevant are the secret key sizes, that are almost half the size in HAETAE compared to DILITHIUM. A direct comparison to FALCON for the same claimed security level is only possible for the highest parameter set, FALCON-1024 has a signature of less than half the size compared to HAETAE-260, and its verification key is about 14% smaller.

Security	120	180	260
q	64513	64513	64513
M	6.0	5.0	6.0
Key Rate	0.1	0.25	0.1
β	354.82	500.88	623.72
B	9388.97	17773.21	22343.66
B'	9382.26	17766.15	22334.95
B''	12320.79	21365.10	24441.49
(k, ℓ)	(2,4)	(3,6)	(4,7)
η	1	1	1
τ	58	80	128
α_{h}	512	512	256
d	1	1	0
Forgery			
BKZ block-size b	409 (333)	617 (512)	878 (735)
Classical hardness	119 (97)	180 (149)	256 (214)
Quantum hardness	105 (85)	158 (131)	225 (188)
Key-Recovery			
BKZ block-size b	428	810	988
Classical hardness	125	236	288
Quantum hardness	109	208	253

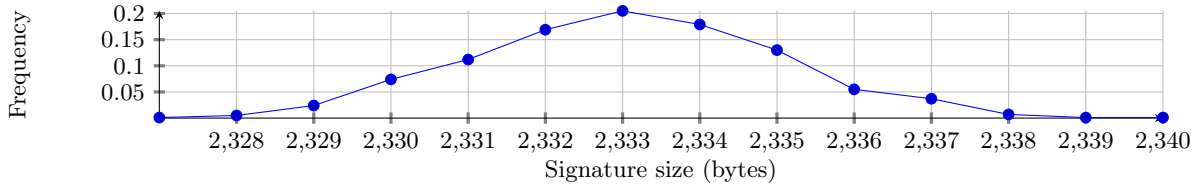
Table 2: HAETAE parameters sets. Hardness is measured with the Core-SVP methodology.

Scheme	Lvl.	vk	Signature	Sum	Secret key
HAETAE-120	2	992	1,463	2,455	1,376
HAETAE-180	3	1,472	2,337	3,809	2,080
HAETAE-260	5	2,080	2,908	4,988	2,720
DILITHIUM-2	2	1,312	2,420	3,732	2,528
DILITHIUM-3	3	1,952	3,293	5,245	4,000
DILITHIUM-5	5	2,592	4,595	7,187	4,864
FALCON-512	1	897	666	1,563	1,281
FALCON-1024	5	1,792	1,280	3,072	2,305

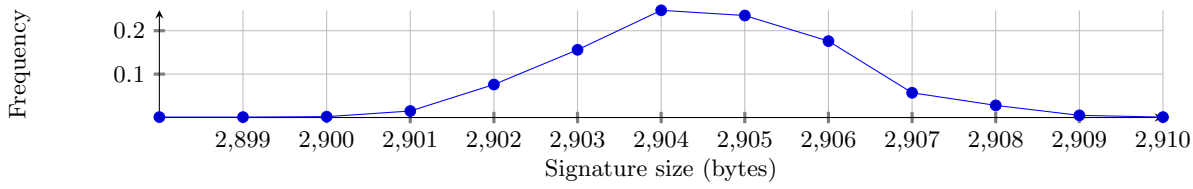
Table 3: NIST security level, signature and key sizes (bytes) of HAETAE, DILITHIUM, and FALCON.



(a) HAETAE-120



(b) HAETAE-180



(c) HAETAE-260

Fig. 9: Signature size distribution over 1000 executions.

4 Implementation

In this section, we give the implementation details of HAETAE. The constant-time reference implementation and the supporting scripts can be found on the team HAETAE website: `https://kpqc.cryptolab.co.kr`.

4.1 Performance

We developed a unoptimized, portable and constant-time implementation in C for HAETAE and report median and average cycle counts of one thousand executions for each parameter set in Table 4. Due to the key and signature rejection steps, the median and average values for key generation and signing respectively differ clearly, whereas the two values are much closer for the verification.

For a fair comparison, we also performed measurements on the same system with identical settings of the reference implementation of DILITHIUM[2] and the implementation with emulated floating-point operations, and thus also fully portable, of FALCON[3], as given in Table 4. The performance of the signature verification for HAETAE is very close to DILITHIUM throughout the parameter sets. HAETAE-180 verification is about 23% slower than its' counter-part, HAETAE-260 on the other hand, is even 6% faster than the respective DILITHIUM parameter set. For key generation and signature computation, our current implementation of HAETAE is clearly slower than DILITHIUM. We measure a slowdown of factors two to five. In comparison to FALCON, however, HAETAE reports 30-90 times faster key generation and 2-4 times faster signing speed. For the verification, FALCON outperforms both DILITHIUM and HAETAE by a factor of four.

A closer look at the key generation reveals that the complex Fast Fourier Transformation that is required for the rejection step, is with 53% by far the most expensive operation and a sensible target for optimized implementations.

Profiling the signature computation reveals that the slowdown compared to DILITHIUM is mainly caused by the sampling from a hyperball, where about 80% of the computation time is spent. The hyperball sampling itself is dominated by the generation of randomness, which we derive from the extendable output function SHAKE256 [Dwo15], which is also used in the DILITHIUM implementation. Almost 60% of the signature computation time is spend in SHAKE256.

Based on the profiling and benchmarking of subcomponents, we estimate the performance of a randomized HAETAE implementation with pre-computation. The generic version, that is independent of the key, would already achieve a speedup of a factor five for its online signing, because the expensive hyperball sampling can be done offline. For the pre-computation variant with a designated signing key, additionally a lot of matrix-vector multiplications and therefore most of the transformations from and to the Number-Theoretic Transform (NTT) domain, can be precomputed. We estimate about 12% of the full deterministic signing running time, for the online signing in this case.

While the smallest parameter set HAETAE-120 yields the fastest implementation, our parameter selection leads to the unusual situation, that the most secure HAETAE-260 is very close or even faster in key generation and signing than HAETAE-180. HAETAE-180 is nevertheless a viable option, due to the smaller signature and key sizes compared to HAETAE-260.

Our rANS encoding is based on an implementation by Fabian Giesen [Gie14].

4.2 Optimized and Embedded Implementation

We provide an optimized implementation using AVX2 featuring

- vectorized NTT and inverse NTT,
- vectorized addition, subtraction, and point-wise multiplication, and
- four-way parallel hashing.

[2] https://github.com/pq-crystals/dilithium/tree/master/ref
[3] https://falcon-sign.info/falcon-round3.zip

Scheme		KeyGen	Sign	Verify
HAETAE-120	*med*	1,403,720	6,336,104	394,132
	ave	1,862,874	9,017,863	394,954
HAETAE-180	*med*	2,383,130	9,604,732	722,602
	ave	3,538,029	11,917,124	724,663
HAETAE-260	*med*	1,710,610	9,165,754	929,354
	ave	2,147,863	12,075,035	930,946
DILITHIUM-2	*med*	339,334	1,140,794	367,264
	ave	339,569	1,446,174	367,399
DILITHIUM-3	*med*	609,696	1,955,296	585,536
	ave	610,114	2,359,859	585,755
DILITHIUM-5	*med*	935,830	2,473,582	975,802
	ave	936,202	2,904,138	976,350
FALCON-512	*med*	53,778,476	17,332,716	103,056
	ave	60,301,272	17,335,484	103,184
FALCON-1024	*med*	154,298,384	38,014,050	224,378
	ave	178,516,059	38,009,559	224,840

Table 4: Median and average cycle counts of 1000 executions for HAETAE, DILITHIUM, and FALCON. Cycle counts were obtained on one core of an Intel Core i7-10700k, with TurboBoost and hyperthreading disabled.

Parameter Set		KeyGen	Sign	Verify
HAETAE-120	*med*	885,978	2,290,490	153,024
	ave	1,197,131	3,386,322	153,851
HAETAE-180	*med*	1,398,640	3,317,768	236,170
	ave	2,141,589	4,088,249	236,918
HAETAE-260	*med*	1,023,946	3,027,608	291,468
	ave	1,404,192	3,970,666	294,646

Table 5: Median and average cycle counts of 1000 executions for an optimized implementation of HAETAE.

Notably, the sampling algorithms (except for hashing) and the FFT are yet subject to optimization. For this implementation, we obtain the performance figures as shown in Table 5. Moreover, we are working on an embedded implementation targeting Cortex-M4. Preliminary results show that the average execution time of signing outperforms FALCON implementations from pqm4.

4.3 Security against physical attacks

Implementation security is a crucial aspect of making cryptosystems feasible in real-world applications. A significant advantage of HAETAE is that it can be protected against power side-channel attacks efficiently and with reasonable overhead. In this context, we emphasize the similarity of HAETAE to DILITHIUM. Hence, past works analyzing concrete attacks [BP18, MUTS22], but also countermeasures [MGTF19, ABC+22], mainly apply to HAETAE as well.

Here, we briefly sketch the feasibility of a side-channel secure implementation. During signing, the most critical operation is multiplying the (public) challenge polynomial c with $\mathbf{s}$ and subsequently adding the result to $\mathbf{y}$. Since this operation may leak information about the secret key statistically over many executions, implementers must protect it accordingly. As countermeasures against these so-called Differential Power Analysis (DPA) attacks, masking has been proven effective.

This operation is straightforward to mask at arbitrary order by splitting the secret key polynomials into multiple additive shares in $\mathcal{R}_q$. A masked implementation then stores the NTT of each share of $\mathbf{s}$ and multiplies them to c, obtaining a shared $c\mathbf{s}$. Following this, the inverse NTT is applied share-wise. Since $\mathbf{y}$ is a polynomial vector in $(1/N)\mathcal{R}$, it is not trivially possible to add our shares of $c\mathbf{s} \in \mathcal{R}_q^{k+\ell}$.

On the other hand, $\mathbf{y}$ is not a secret-key-dependent value. Therefore, it is not required to be protected against DPA but only against the much stronger attacker model of an Simple Power Analysis (SPA). In fact, coefficient-wise shuffling of the addition is sufficient at this point (cf. [ABC+22]). This might involve a masking conversion from $\mathbb{Z}_q$ to $\mathbb{Z}_{2^{32}}$, but no multiplication of masked fix-point values, which would be costly. Subsequently, the computation of $2\mathbf{z} - \mathbf{y}$ and the bound checks can be shuffled without applying costly masking.

The same idea applies to the whole hyperball sampling procedure. Since the order of the Gaussian samples is, in principle, irrelevant, they can be generated in random order. This is particularly an advantage for randomized HAETAE.

It is noteworthy that hashing the challenge seed is only required to be protected against SPA as well. Since the input order into the hash function cannot be randomized, the preceding values must still be protected by masking. Therefore, we propose to perform a shuffled point-wise multiplication of $\mathbf{A}$ and $\mathbf{y}$, directly followed by freshly masking the resulting coefficients. Then, a share-wise inverse NTT and a masking conversion to the Boolean domain will be performed, which enables a secure HighBits operation. For the LSBs of y_0, generating a fresh Boolean masking during the shuffled generation of the hyperball sample's coefficients is sufficient.

Comparison to FALCON and MITAKA While there is no known method to efficiently mask FALCON, MITAKA [EFG+22b] was designed to be easy to protect against implementation attacks, while still having the advantage of similarly small signatures as FALCON. For MITAKA, the crux regarding side-channel security is sampling Gaussian-distributed values. Together with MITAKA, an efficient, masked algorithm for discrete Gaussian sampling was presented. However, Prest broke its security proof recently [Pre23]. In this respect, HAETAE has the strong advantage, that Gaussian sampling only needs to be secured against the much stronger SPA attacker model, which allows for simpler countermeasures, while MITAKA's side-channel security will always depend on a masked sampler.

4.4 Hardware implementations

Hashing and generation of randomness are the most time-consuming operations of HAETAE. Therefore, we assume that hardware implementations will bring significant speedup and can be competitive to

DILITHIUM, particularly through efficient KECCAK cores. Furthermore, hardware implementations will benefit significantly from applying the offline approach. Naturally, a module generating hyperball samples can be instantiated and run parallel to the online phase, thus, hiding its latency behind the online phase. Moreover, high-speed applications could adopt the offline approach with designated signing key, including the multiplication of $\mathbf{A}$ and $\mathbf{y}$, to further reduce the latency of the online phase.

5 Security

Unforgeability under Chosen Message Attacks (UF-CMA) is regarded as a standard security notion for digital signature schemes. The adversary is given the verification key and has access to a signing oracle to call on (adaptively) chosen messages. The adversary wins if it forges a valid signature of a new, non-queried message. *Strong Unforgeability under Chosen Message Attacks* (SUF-CMA) is a slightly stronger security notion than UF-CMA: the adversary wins if it forges a valid signature-message pair that it did not already see.

The concrete SUF-CMA security of HAETAE can be proven in the classical Random Oracle Model (ROM) under the standard MLWE and MSIS assumptions. However, since the proof is based on the forking lemma, the reduction is not tight, and it is not applicable in the Quantum Random Oracle Model (QROM) setting. First, using the zero-knowledge property of the underlying identification scheme, *Unforgeability under No Message Attacks* (UF-NMA) reduces to (S)UF-CMA security, both in the ROM and the QROM.

As pointed out in [DFPS23] and [BBD+23], the security proofs in [AFLT16, KLS18] contain flaws. However, these works also introduce fixes. We therefore base our security proof on the fixed analyses of both [DFPS23] and [BBD+23].

UF-NMA is directly related to a problem that can be viewed as a "convolution" of lattice and hash function problems. We call this problem BimodalSelfTargetMSIS. Similar to the SelfTargetMSIS described in [DKL+18, KLS18], we can analyze the UF-CMA security based on the MLWE and BimodalSelfTargetMSIS assumptions. Note that in the ROM, MSIS reduces to BimodalSelfTargetMSIS, but the reduction is not tight and does not readily extend to quantum adversaries (it relies on the forking lemma). This said, this non-tightness and limitation to classical adversaries is not known to reflect any weakness.

For setting parameters, we consider the hardness of MSIS and MLWE for relevant parameters. Intuitively, the MLWE assumption is used for security against key-recovery attacks, and the BimodalSelfTargetMSIS used for security against forgeries is identified to the MSIS assumption.

5.1 Security definition

We introduce the BimodalSelfTargetMSIS assumption and give a classical reduction from the standard MSIS assumption. BimodalSelfTargetMSIS is a variant of the SelfTargetMSIS assumption adapted to the bimodal setup.

Definition 6 ($\mathsf{BimodalSelfTargetMSIS}_{H,n,q,k,\ell,\beta}$)**.** *Suppose that $H : \{0,1\}^* \times \mathcal{M} \to \mathcal{R}_2$ is a cryptographic hash function. For positive integers q, k, ℓ, a positive real number β and the dimension n of $\mathcal{R}$, we say that the advantage of an adversary $\mathcal{A}$ solving the search-*$\mathsf{BimodalSelfTargetMSIS}_{H,n,q,k,\ell,\beta}$ *problem with respect to $\mathbf{j} \in \mathcal{R}_2^k \setminus \{0\}$ is*

$$\mathsf{Adv}^{\mathsf{BimodalSelfTargetMSIS}}_{H,n,q,k,\ell,\beta}(\mathcal{A}) = \Pr\left[\begin{array}{c|c} 0 < \|\mathbf{y}\|_2 < \beta \wedge & (\mathbf{A}_0, \mathbf{b}) \leftarrow \mathcal{R}_q^{k\times(\ell-1)} \times \mathcal{R}_q^k; \\ H(\mathbf{A}\mathbf{y} - qc\mathbf{j} \mod 2q, M) = c & \mathbf{A} = (-2\mathbf{b} + q\mathbf{j} \mid 2\mathbf{A}_0 \mid 2\mathbf{Id}_k) \mod 2q; \\ & (\mathbf{y}, c, M) \leftarrow \mathcal{A}^{|H(\cdot)\rangle}(\mathbf{A}) \end{array}\right].$$

In the ROM (resp. QROM), the adversary is given classical (resp. quantum) access to H.

Theorem 2 (Classical Reduction from MSIS **to** BimodalSelfTargetMSIS**).** *Assume that q is odd, $H : \{0,1\}^* \times \mathcal{M} \to \mathcal{R}_2$ is a cryptographic hash function modeled as a random oracle and that every polynomial-time classical algorithm has a negligible advantage against $\mathsf{MSIS}_{n,q,k,\ell,\beta}$. Then every polynomial-time classical algorithm has negligible advantage against $\mathsf{BimodalSelfTargetMSIS}_{n,q,k,\ell,\beta/2}$.*

Proof *(sketch).* Consider a $\mathsf{BimodalSelfTargetMSIS}_{n,q,k,\ell,\beta/2}$ classical algorithm $\mathcal{A}$ that is polynomial-time and has classical access to H. If $\mathcal{A}^{|H(\cdot)\rangle}(\mathbf{A})$ makes Q hash queries $H(\mathbf{w}_i, M_i)$ for $i = 1, \cdots, Q$ and outputs a solution $(\mathbf{y}, c, M_j)$ for some $j \in [Q]$, then we can construct an adversary $\mathcal{A}'$ for $\mathsf{MSIS}_{n,q,k,\ell,\beta}$ as follows.

The adversary $\mathcal{A}'$ can first rewind $\mathcal{A}$ to the point at which the i-th query was made and reprogram the hash as $H(\mathbf{w}_j, M_j) = c'(\neq c)$. Then, with probability approximately $1/Q$, algorithm $\mathcal{A}$ will produce another solution $(\mathbf{y}', c', M_j)$. We then have

$$\mathbf{A}\mathbf{y} - qc\mathbf{j} = \mathbf{z}_j = \mathbf{A}\mathbf{y}' - qc'\mathbf{j} \bmod 2q \quad \text{and} \quad \|\mathbf{y}\|_2, \|\mathbf{y}\|_2 < \beta/2.$$

As q is odd, we have $\mathbf{A}(\mathbf{y} - \mathbf{y}') = (c - c')\mathbf{j} \bmod 2$. The fact that $c' \neq c$ implies that the latter is non-zero modulo 2, and hence so is $\mathbf{y} - \mathbf{y}'$ over the integers. As it also satisfies $(-\mathbf{b}|\ \mathbf{A}_0|\ \mathbf{Id}_k) \cdot (\mathbf{y} - \mathbf{y}') = 0 \bmod q$ and $\|\mathbf{y} - \mathbf{y}'\| < \beta$, it provides a $\mathsf{MSIS}_{n,q,k,\ell,\beta}$ solution for the matrix $(-\mathbf{b}|\ \mathbf{A}_0|\ \mathbf{Id}_k)$, where the submatrix $(-\mathbf{b}|\ \mathbf{A}_0) \in \mathcal{R}_q^{k \times \ell}$ is uniform. □

The above classical reduction from MSIS to BimodalSelfTargetMSIS is very similar to the reduction from MSIS to SelfTargetMSIS introduced in [DKL+18] and is similarly non-tight. Moreover, since the reduction relies on the forking lemma; it cannot be directly extended to a quantum reduction in the QROM.

Security definitions. We recall the definitions of the above security notions for digital signatures.

Definition 7 (Unforgeability under No Message Attacks (UF-NMA)). *For a signature scheme $\mathcal{S} =$ (KeyGen, Sign, Verify), the advantage of a UF-NMA adversary $\mathcal{A}$ is defined as:*

$$\mathsf{Adv}_{\mathcal{S}}^{\mathsf{UF\text{-}NMA}}(\mathcal{A}) = \Pr\left[\mathsf{Verify}(\mathsf{vk}, M, \sigma) = 1 |\ (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}; (M, \sigma) \leftarrow \mathcal{A}(\mathsf{vk})\right].$$

Definition 8 (Unforgeability under Chosen Message Attacks (UF-CMA)). *Let $\mathcal{S} =$ (KeyGen, Sign, Verify) be a signature scheme. A UF-CMA adversary $\mathcal{A}$ has access to the verification key and a signing oracle to make adaptive queries. Let the queried messages and the received signatures be (M_i, σ_i) for $i = 1, \cdots, Q$. At the end of the experiment, it outputs a message-signature pair (M^*, σ^*). Then the advantage of $\mathcal{A}$ is defined as:*

$$\mathsf{Adv}_{\mathcal{S}}^{\mathsf{UF\text{-}CMA}}(\mathcal{A}) = \Pr\left[\begin{matrix} M^* \notin \{M_i\}_{i\in[Q]} \ \wedge \\ \mathsf{Verify}(\mathsf{vk}, M^*, \sigma^*) = 1 \end{matrix} \middle| \begin{matrix} (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}; \\ (M^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)} \end{matrix}\right].$$

Definition 9 (Strong Unforgeability under Chosen Message Attacks (SUF-CMA)). *Let $\mathcal{S} =$ (KeyGen, Sign, Verify) be a signature scheme. An SUF-CMA adversary $\mathcal{A}$ has access to the verification key and a signing oracle to make adaptive queries. Let the queried messages and the received signatures be (M_i, σ_i) for $i = 1, \cdots, Q$. At the end of the experiment, it outputs a message-signature pair (M^*, σ^*). Then the advantage of $\mathcal{A}$ is defined as:*

$$\mathsf{Adv}_{\mathcal{S}}^{\mathsf{SUF\text{-}CMA}}(\mathcal{A}) = \Pr\left[\begin{matrix} (M^*, \sigma^*) \notin \{(M_i, \sigma_i)\}_{i\in[Q]} \\ \wedge\ \mathsf{Verify}(\mathsf{vk}, M^*, \sigma^*) = 1 \end{matrix} \middle| \begin{matrix} (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}; \\ (M^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)} \end{matrix}\right].$$

HAETAE achieves UF-CMA security in (Q)ROM, assuming MLWE and BimodalSelfTargetMSIS are hard.

Theorem 3 (UF-CMA Security of HAETAE in the QROM). HAETAE *in Figure 7 is* UF-CMA *secure in the QROM.*

Proof *(sketch).* The proof relies on the analysis of [DFPS23], which reduces UF-CMA security to UF-NMA security, where an adversary is not allowed to make signing queries. This analysis requires that the commitment min-entropy is high and the underlying Σ-protocol is Honest-Verifier Zero-Knowledge (HVZK). The latter is proved by providing a simulator for non-aborting transcripts and proving that the distribution of $\lfloor \mathbf{y} \rceil$ has sufficiently large min-entropy.

Commitment min-entropy. We first claim that the underlying Σ-protocol has large commitment min-entropy. The underlying identification protocol has ε *bits of min-entropy* if

$$\forall(\mathbf{w}, x), \Pr_{\mathbf{y}}\left[(\mathsf{HighBits}^{\mathsf{h}}(\mathbf{A}\lfloor\mathbf{y}\rceil), \mathsf{LSB}(\lfloor y_0 \rceil)) = (\mathbf{w}, x)\right] \leq 2^{-\varepsilon},$$

for any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$ and $\mathbf{y} \leftarrow U(\mathcal{B}_{(1/N)\mathcal{R},(k+\ell)}(B))$. We note that $\mathsf{LSB}(\lfloor y_0 \rceil)$ is a binary vector of length n and is statistically close to uniform. Thus, the inner probability is (very loosely) bounded by 2^{-n} regardless of the choice of $(\mathsf{pk}, \mathsf{sk})$. Hence we obtain at least 256 bits of min-entropy in all of our parameter sets.

HVZK. Next, we show that the underlying Σ-protocol satisfies the HVKZ property. To do so, we follow the strategy from [DFPS23, Section 4.2], which studies the simulation of non-aborting transcripts and switch to computational mode for aborting ones. We propose the following simulator in Figure 10. On input a challenge c, it runs $\mathcal{A}(0)$ as defined in Figure 2 (note that we are prevented from using $\mathcal{B}$ as the exact rejection probability is unknown), and if it fails, it samples a uniform commitment and no answer. Here, $p(\mathbf{z})$ is $1/2$ if $\|\mathbf{z}\| \leq r$ and 0 everywhere else.

$\mathsf{Sim}(\mathbf{A}, c)$:
1: $\mathbf{y} \leftarrow U(\mathcal{B}_{(1/N)\mathcal{R},m}(r'))$
2: $\mathbf{w} \leftarrow (\mathsf{HighBits}^{\mathsf{h}}(\mathbf{A}\lfloor\mathbf{y}\rceil), \mathsf{LSB}(y_0))$
3: $\mathbf{z} \leftarrow \mathbf{y}$
4: $\mathbf{u} \leftarrow U(\mathcal{R}_q^k)$
5: $u_0 \leftarrow U(\mathcal{R}_2)$
6: $\tilde{\mathbf{w}} \leftarrow (\mathsf{HighBits}^{\mathsf{h}}(2\mathbf{u} + q\mathbf{j}u_0), u_0)$
7: **return** $(\mathbf{w}, c, \mathbf{z})$ with probability $p(\mathbf{z})$, **else** $(\tilde{\mathbf{w}}, c, \perp)$

Fig. 10: HAETAE transcript simulator

(i) Simulating non-aborting transcripts. When a sample is accepted, Lemma 1 states that the simulator follows exactly the same distribution as the real aglorithm.

(ii) Simulating aborting transcripts. As argued in [DFPS23, Section 4.2], in this context, we can use a a computational notion of HVZK rather than the usual statistical definition. We introduce an LWE-like assumption which states that it is hard to distinguish $\mathbf{w} = \mathbf{A}\lfloor\mathbf{y}\rceil \bmod q$ from a uniform element mod$q$. This LWE assumption is unusual only in its choice of distribution for the noise and the secret.

These two properties allow us to apply [DFPS23, Theorem 4] to reduce the SUF-CMA security to UF-NMA security.

If one wants to avoid this assumption, it is possible to use the reduction from [BBD+23] by using $\mathcal{A}(0)$ as a simulator. The non-aborting transcripts produced by this simulator have statistical distance 0 with real ones.

Proving UF-NMA security. Finally, we note that the UF-NMA security game is exactly the problem defined in Definition 6, up to replacing the verification key by an uniform matrix (still in HNF form), which is done under the MLWE assumption.

5.2 Cost of known attacks

For the concrete security analysis, we list the best known lattice attacks and consider their costs for attacking HAETAE.

All the best known attacks rely on the Block–Korkine–Zolotarev (BKZ) lattice reduction algorithm [SE94, CN11, HPS11]. The BKZ algorithm is a lattice basis reduction algorithm that repeatedly uses a Shortest Vector Problem (SVP) solver in small-dimensional projected sublattices. The dimension b of these projected

sublattices is called the block-size. BKZ with block-size b hence relies on an SVP solver in dimension b. The block-size drives the cost of BKZ and determines the resulting basis's quality. It provides a quality/time trade-off: If b gets larger, better quality will be guaranteed, but the time complexity for the SVP solver will exponentially increase. The time complexity of the b-BKZ algorithm is the same as the SVP solver for dimension b, up to polynomial factors. Hence the time complexity differs depending on the SVP solver used. The most efficient SVP algorithm uses the sieving method proposed by Becker et al. [BDGL16] which takes time $\approx 2^{0.292b+o(b)}$. The fastest known quantum variant is proposed by Chailloux and Loyer in [CL21] and takes time $\approx 2^{0.257b+o(b)}$.

Based on the BKZ algorithm, we will follow the *core-SVP* methodology from [ADPS16] and as in the subsequent lattice-based schemes [ABB+19, DKL+18, FHK+17, DKSRV18, BDK+18]. It is regarded as a conservative way to set security parameters. We ignore the polynomial factors and the $o(b)$ terms in the exponents of the run-time bounds above for the time complexity of the BKZ algorithm.

We consider the *primal attack* and the *dual attack* for MLWE, and the plain BKZ attack for MSIS and BimodalSelfTargetMSIS problems. We remark that any $\mathsf{MLWE}_{n,q,k,\ell,\eta}$ instance can be viewed as an $\mathsf{LWE}_{q,nk,n\ell,\eta}$ instance, and also any $\mathsf{MSIS}_{n,q,k,\ell,\beta}$ can be viewed as an $\mathsf{SIS}_{q,nk,n\ell,\beta}$ instance. Even though the MLWE and MSIS problems have some extra algebraic structure compared to the LWE and SIS problems, we do not currently know how to exploit it to improve the best known attacks. For this reason, we estime the concrete hardness of the MLWE and MSIS problems over the structured lattices as the concrete hardness of the corresponding LWE and SIS problems over the unstructured lattices.

We summarize the costs of the known attacks in Table 6. In the table, the required block-sizes for BKZ and the costs of the attacks in core-SVP hardness are given, estimated by the python script we submitted to the KpqC competition with this document. It is a modification of the security estimator of Dilithium [DS20]. The parameters for MLWE and MSIS problems are chosen based on Theorems 2 and 3. The numbers in parentheses are for the SUF-CMA security of randomized HAETAE (in the case of the deterministic signature, strong and weak unforgeability are the same). All costs are rounded downwards.

Parameter sets	HAETAE120	HAETAE180	HAETAE260
Target security	120	180	260
BKZ block-size b to break SIS	409 (333)	617 (512)	878 (735)
Classical hardness	119 (97)	180 (149)	256 (214)
Quantum hardness	105 (85)	158 (131)	225 (188)
BKZ block-size b for primal attack	431	820	1001
Classical hardness	126	239	292
Quantum hardness	110	210	257
BKZ block-size b for dual attack	428	810	988
Classical hardness	125	236	288
Quantum hardness	109	208	253

Table 6: Core-SVP hardness for the best known attacks

Primal attack. Given an LWE instance $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{k\times\ell} \times \mathbb{Z}_q^k$, we first define the lattices $\Lambda_m = \{\mathbf{v} \in \mathbb{Z}^{\ell+m+1} : \mathbf{Bv} = \mathbf{0} \mod q\}$ for all $m \leq k$, where $\mathbf{B} = \left(\mathbf{A}_{[m]} | \mathbf{Id}_m | \mathbf{b}_{[m]}\right) \in \mathbb{Z}_q^{m\times(\ell+m+1)}$, $\mathbf{A}_{[m]}$ is the uppermost $m \times \ell$ sub-matrix of $\mathbf{A}$ and $\mathbf{b}_{[m]}$ is the uppermost m-dimensional sub-vector of $\mathbf{b}$. As $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{k\times\ell} \times \mathbb{Z}_q^k$ is an LWE instance, there exist $\mathbf{s}$ and $\mathbf{e}$ short such that $\mathbf{b} = \mathbf{As} + \mathbf{e}$. This implies that $(\mathbf{s}| \mathbf{e}| -1)$ is a short vector of Λ_m. The primal attack consists in running BKZ on Λ_m to find short vectors in Λ_m. The variable m is optimized to minimize the cost of the attack.

Dual attack. Given an LWE instance $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{k\times\ell} \times \mathbb{Z}_q^k$, we first define the lattices $\Lambda'_m = \{(\mathbf{u}, \mathbf{v}) \in \mathbb{Z}^m \times \mathbb{Z}^\ell \ : \ \mathbf{A}_{[m]}^\top \mathbf{u} + \mathbf{v} = \mathbf{0} \mod q\}$ for all $m \leq k$, where $\mathbf{A}_{[m]}$ is the uppermost $m \times \ell$ sub-matrix of $\mathbf{A}$. If $(\mathbf{u}, \mathbf{v})$ is a short vector in Λ'_m, then $\mathbf{u}^\top\mathbf{b} = \mathbf{v}^\top\mathbf{s} + \mathbf{u}^\top\mathbf{e}_{[m]}$ is short if $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$ for short vectors $\mathbf{s}$ and $\mathbf{e}$, and is uniformly distributed modulo q if $\mathbf{b}$ is uniform and independent from $\mathbf{A}$ (here $\mathbf{e}_{[m]}$ refers to the uppermost m-dimensional sub-vector of $\mathbf{e}$). This provides a distinguishing attack. The dual attack consists in finding a short non-zero vector in the lattice Λ'_m using BKZ. The variable m is optimized to minimize the cost of the attack.

SIS attack. To analyze the hardness of BimodalSelfTargetMSIS, we analyze the hardness of the corresponding MSIS. Intuitively, if we assume that H is a cryptographic hash, then the input structure will not help find the preimage. So we can assume that M is fixed. Then the problem turns into finding the preimage $\mathbf{x}$ of c with respect to $H(\cdot, M)$ and then finding $\mathbf{y}$ satisfying $\mathbf{x} = \mathbf{A}\mathbf{y} - qc\mathbf{j} \mod 2q$. Apart from the first step, if we have the preimage c, then the second step will be turned into finding $\mathbf{y}'$ satisfying $(2\mathbf{b}|\ \mathbf{A}_0|\ \mathbf{Id}_k) \cdot \mathbf{y}' = \mathbf{t} \mod q$, for a known vector $\mathbf{t}$ over $\mathcal{R}_q$. Here, $\mathbf{y}'$ is defined as $\mathbf{y}' = ((y_0 - x'_0)/2, y_1, \cdots, y_{k+\ell-1})^\top$ and $\mathbf{t} = 2^{-1} \cdot \mathbf{x} + x'_0\mathbf{b}$ mod q, where $x'_0 = (x_0 + c \mod 2)$ which actually decides the LSB of y_0. Also, $\|\mathbf{y}'\|_2$ is bounded by the same bound used for $\|\mathbf{y}\|_2$. This implies that solving BimodalSelfTargetMSIS is at least as hard as solving MSIS with the same norm bound or finding the preimage of a hash as an attack perspective. However, for the hardness of BimodalSelfTargetMSIS problem, we will analyze it more conservatively, as the hardness of MSIS problem with a twice larger norm bound, taking into account the classical reduction from MSIS to BimodalSelfTargetMSIS in Theorem 2. We analyze the best known attacks for SIS problem, for both MSIS and BimodalSelfTargetMSIS problems that the unforgeability of our signature scheme relies on.

Given an SIS instance $\mathbf{A} \in \mathbb{Z}_q^{k\times\ell}$ with a bound β, we define the lattices $\Lambda''_m = \{\mathbf{u} \in \mathbb{Z}^m \ : \ \mathbf{B}\mathbf{u} = \mathbf{0} \mod q\}$ for all $m \leq k + \ell$, where $\mathbf{B}$ is the $k \times m$ leftmost sub-matrix of $(\mathbf{A}|\ \mathbf{Id}_k)$. Then a short non-zero vector in the lattice Λ''_m is a solution to the SIS problem. Once more, we use BKZ and optimize the choice of m.

Note that if $\beta > q$, then there are some trivial non-zero solutions to SIS problem such as $(q, 0, \cdots, 0)$ with ℓ_2-norm $< \beta$. Depending on the parameters, the security could be affected by some existing attacks [DKL+18]. We choose the prime q larger than the MSIS bound β to avoid such weaknesses.

References

ABB+19. Erdem Alkim, Paulo S. L. M. Barreto, Nina Bindel, Juliane Kramer, Patrick Longa, and Jefferson E. Ricardini. The lattice-based digital signature scheme qTESLA. Cryptology ePrint Archive, Number 2019/085, 2019. https://eprint.iacr.org/2019/085.

ABC+22. Melissa Azouaoui, Olivier Bronchain, Gaëtan Cassiers, Clément Hoffmann, Yulia Kuzovkova, Joost Renes, Markus Schönauer, Tobias Schneider, François-Xavier Standaert, and Christine van Vredendaal. Leveling Dilithium against leakage: Revisited sensitivity analysis and improved implementations. Cryptology ePrint Archive, Report 2022/1406, 2022. https://eprint.iacr.org/2022/1406.

ADPS16. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key Exchange—A New Hope. In *25th USENIX Security Symposium*, pages 327–343. USENIX Association, 2016.

AFLT16. Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly secure signatures from lossy identification schemes. *J. Cryptol.*, 29(3):597–631, 2016.

BBD+23. Manuel Barbosa, Gilles Barthe, Christian Doczkal, Jelle Don, Serge Fehr, Benjamin Grégoire, Yu-Hsuan Huang, Andreas Hülsing, Yi Lee, and Xiaodi Wu. Fixing and mechanizing the security proof of Fiat-Shamir with aborts and Dilithium. Cryptology ePrint Archive, Paper 2023/246, 2023. https://eprint.iacr.org/2023/246.

BBE+19. Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Mélissa Rossi, and Mehdi Tibouchi. GALACTICS: gaussian sampling for lattice-based constant- time implementation of cryptographic signatures, revisited. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of CCS*, pages 2147–2164. ACM, 2019.

BDGL16. Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 10–24, 2016.

BDK+18. Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367, 2018.

BG14. Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In Josh Benaloh, editor, *Topics in Cryptology – CT-RSA 2014*, pages 28–47, 2014.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science (ITCS)*, pages 309–325. ACM, 2012.

BP18. Leon Groot Bruinderink and Peter Pessl. Differential fault attacks on deterministic lattice signatures. *IACR TCHES*, 2018(3):21–43, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7267.

CL21. André Chailloux and Johanna Loyer. Lattice sieving via quantum random walks. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT*, pages 63–91. Springer, 2021.

CN11. Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT*, pages 1–20. Springer, 2011.

DDLL13. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO*, pages 40–56. Springer, 2013.

DFPS22. Julien Devevey, Omar Fawzi, Alain Passelègue, and Damien Stehlé. On rejection sampling in lyubashevsky's signature scheme. In *Asiacrypt*. Springer, 2022.

DFPS23. Julien Devevey, Pouria Fallahpour, Alain Passelègue, and Damien Stehlé. A detailed analysis of Fiat-Shamir with aborts. Cryptology ePrint Archive, Paper 2023/245, 2023. https://eprint.iacr.org/2023/245.

DKL+18. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, 2018.

DKSRV18. Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In *Africacrypt*, pages 282–305. Springer, 2018.

DS20. Léo Ducas and John Schanck. Security estimation scripts for kyber and dilithium, 2020. GitHub repository, available at https://github.com/pq-crystals/security-estimates.

Dud13. Jarek Duda. Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding, 2013. ArXiv preprint, available at https://arxiv.org/abs/1311.2540.

Dwo15. Morris J Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions. 2015.

EFG+22a. Thomas Espitau, Pierre-Alain Fouque, François Gérard, Mélissa Rossi, Akira Takahashi, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Mitaka: A simpler, parallelizable, maskable variant of Falcon. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT*, pages 222–253. Springer, 2022.

EFG+22b. Thomas Espitau, Pierre-Alain Fouque, François Gérard, Mélissa Rossi, Akira Takahashi, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Mitaka: A simpler, parallelizable, maskable variant of falcon. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 222–253. Springer, Heidelberg, May / June 2022.

EFGT17. Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. In Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1857–1874, 2017.

ETWY22. Thomas Espitau, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Shorter hash-and-sign lattice-based signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO*, pages 245–275. Springer, 2022.

FHK+17. Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-Fourier lattice-based compact signatures over NTRU, 2017. Submission to the NIST post-quantum cryptography standardization process.

Gie14. Fabian Giesen. Interleaved entropy coders. *arXiv preprint arXiv:1402.3392*, 2014.

HPS11. Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Algorithms for the shortest and closest lattice vector problems. In Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology*, pages 159–190. Springer, 2011.

KLS18. Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In *Advances in Cryptology – EUROCRYPT*, pages 552–586. Springer, 2018.

LS15. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.*, 75(3):565–599, 2015.

Lyu09. Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT*, pages 598–616. Springer, 2009.

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT*, pages 738–755. Springer, 2012.

MGTF19. Vincent Migliore, Benoît Gérard, Mehdi Tibouchi, and Pierre-Alain Fouque. Masking Dilithium - efficient implementation and side-channel evaluation. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 344–362. Springer, Heidelberg, June 2019.

MUTS22. Soundes Marzougui, Vincent Ulitzsch, Mehdi Tibouchi, and Jean-Pierre Seifert. Profiling side-channel attacks on Dilithium: A small bit-fiddling leak breaks it all. Cryptology ePrint Archive, Report 2022/106, 2022. `https://eprint.iacr.org/2022/106`.

Pre17. Thomas Prest. Sharper bounds in lattice-based cryptography using the Rényi divergence. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT*, pages 347–374. Springer, 2017.

Pre23. Thomas Prest. A key-recovery attack against mitaka in the t-probing model. Cryptology ePrint Archive, Report 2023/157, 2023. `https://eprint.iacr.org/2023/157`.

SE94. Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1):181–199, 1994.

VGS17. Aaron R Voelker, Jan Gosmann, and Terrence C Stewart. Efficiently sampling vectors and coordinates from the n-sphere and n-ball. *Centre for Theoretical Neuroscience-Technical Report*, 01 2017.

A Uncompressed HAETAE

In this appendix, we present HAETAE without its compression step. Readers who are not familiar with the Fiat-Shamir with Aborts line of work may find it easier to read this version first. It highlights the use of bimodal rejection sampling applied to the Fiat-Shamir with Aborts paradigm.

The key generation algorithms ensures that $\mathbf{As} = q\mathbf{j} \bmod 2q$, while also putting $\mathbf{A}$ in a "close to Hermite Normal Form". Namely, instead of the right part of $\mathbf{A}$ being $\mathbf{Id}_k$, it is $2\mathbf{Id}_k$. This subtlety impacts the compression design, in the uncompressed version of HAETAE.

The signature for a message M consists of $c = H(\mathbf{A}\lfloor \mathbf{y} \rceil \bmod 2q, M)$ and $\mathbf{z} = \lfloor \mathbf{y} \rceil \pm c\mathbf{s}$. Sometimes, the vector $\mathbf{z}$ is rejected and the signing procedure is restarted. Note that $\mathbf{Az} = \mathbf{A}\lfloor \mathbf{y} \rceil + qc\mathbf{j} \bmod 2q$, independently of the sign that was chosen for $c\mathbf{s}$. The verification step then checks the consistency of the pair $(\mathbf{z}, c)$ and the smallness of $\mathbf{z}$.

$\underline{\mathsf{KeyGen}(1^\lambda)}$

1: $\mathbf{A}_{\mathsf{gen}} \leftarrow \mathcal{R}_q^{k\times(\ell-1)}$ and $(\mathbf{s}_{\mathsf{gen}}, \mathbf{e}_{\mathsf{gen}}) \leftarrow S_\eta^{\ell-1} \times S_\eta^k$
2: $\mathbf{b} = \mathbf{A}_{\mathsf{gen}} \cdot \mathbf{s}_{\mathsf{gen}} + \mathbf{e}_{\mathsf{gen}} \in \mathcal{R}_q^k$
3: $\mathbf{A} = (-2\mathbf{b} + q\mathbf{j} \mid 2\mathbf{A}_{\mathsf{gen}} \mid 2\ \mathbf{Id}_k)$
4: $\mathbf{s} = (1, \mathbf{s}_{\mathsf{gen}}^\top, \mathbf{e}_{\mathsf{gen}}^\top)^\top$
5: **if** $f(\mathbf{s}) > n\beta^2/\tau$ then restart
6: return $\mathsf{sk} = (\mathbf{A}, \mathbf{s})$ and $\mathsf{vk} = \mathbf{A}$

$\underline{\mathsf{Sign}(\mathsf{sk}, M)}$

1: $\mathbf{y} \leftarrow U(\mathcal{B}_{(1/N)\mathcal{R},(k+\ell)}(B))$
2: $\mathbf{w} \leftarrow \mathbf{A}\ \lfloor \mathbf{y} \rceil$
3: $c = H(\mathbf{w}, M) \in \mathcal{R}_2$
4: $\mathbf{z} = \mathbf{y} + (-1)^b c \cdot \mathbf{s}$ for $b \leftarrow U(\{0,1\})$
5: **if** $\|\mathbf{z}\|_2 > B'$, then restart
6: **else if** $\|2\mathbf{z} - \mathbf{y}\|_2 < B$, then restart with probability $1/2$
7: return $\sigma = (\lfloor \mathbf{z} \rceil, c)$

$\underline{\mathsf{Verify}(\mathsf{vk}, M, \sigma = (\tilde{\mathbf{z}}, c))}$

1: $\tilde{\mathbf{w}} = \mathbf{Az} - qc\mathbf{j} \mod 2q$
2: return $(\ c = H(\tilde{\mathbf{w}}, M)\ ) \wedge \left(\ \|\tilde{\mathbf{z}}\| < B + \frac{\sqrt{n(k+\ell)}}{2}\ \right)$

Fig. 11: High-level description of uncompressed HAETAE

B Discretizing Hyperballs

B.1 Useful Lemma

We will rely on the following claim.

Lemma 10. *Let n be the degree of $\mathcal{R}$. Let $m, N, r > 0$ and $\mathbf{v} \in \mathcal{R}^m$. Then the following statements hold:*

1. $|(1/N)\mathcal{R}^m \cap \mathcal{B}_{\mathcal{R},m}(r)| = |\mathcal{R}^m \cap \mathcal{B}_{\mathcal{R},m}(Nr)|$,
2. $|\mathcal{R}^m \cap \mathcal{B}_{\mathcal{R},m}(r, \mathbf{v})| = |\mathcal{R}^m \cap \mathcal{B}_{\mathcal{R},m}(r)|$,
3. $\mathsf{Vol}(\mathcal{B}_{\mathcal{R},m}(r - \sqrt{mn}/2)) \le |\mathcal{R}^m \cap \mathcal{B}_{\mathcal{R},m}(r)| \le \mathsf{Vol}(\mathcal{B}_{\mathcal{R},m}(r + \sqrt{mn}/2))$.

Proof. For the first statement, note that we only scaled $(1/N)\mathcal{R}^m$ and $\mathcal{B}_{\mathcal{R},m}(r)$ by a factor N. For the second statement, note that the translation $\mathbf{x} \mapsto \mathbf{x} - \mathbf{v}$ maps $\mathcal{R}^m$ to $\mathcal{R}^m$.

We now prove the third statement. For $x \in \mathcal{R}^m$, we define $T_{\mathbf{x}}$ as the hypercube of $\mathcal{R}_{\mathbb{R}}^m$ centered in $\mathbf{x}$ with side-length 1. Observe that the $T_{\mathbf{x}}$'s tile the whole space when $\mathbf{x}$ ranges over $\mathcal{R}^m$ (the way bounderies are handled does not matter for the proof). Also, each of those tiles has volume 1. As any element in $T_{\mathbf{x}}$ is at Euclidean distance at most $\sqrt{mn}/2$ from $\mathbf{x}$, the following inclusions hold:

$$\mathcal{B}_{\mathcal{R},m}(r - \sqrt{mn}/2) \subseteq \cup_{\mathbf{x} \in \mathcal{R}^m \cap \mathcal{B}_{\mathcal{R},m}(r)} T_{\mathbf{x}} \subseteq \mathcal{B}_{\mathcal{R},m}(r + \sqrt{mn}/2).$$

Taking the volumes gives the result. □

B.2 Proof of Lemma 1

Proof. Figure 2 is the bimodal rejection sampling algorithm applied to the source distribution $U((1/N)\mathcal{R}^m \cap \mathcal{B}_{\mathcal{R},m}(r'))$ and target distribution $U((1/N)\mathcal{R}^m \cap \mathcal{B}_{\mathcal{R},m}(r))$ (see, e.g., [DFPS22]). For the result to hold, it suffices that the support of the shift of the source distribution by $\mathbf{v}$ is contained in the support of the target distribution. This is implied by $r' \geq \sqrt{r^2 + t^2}$.

We now consider the number of expected iterations, i.e., the maximum ratio between the two distributions. To guide the intuition, note that if we were to use continuous distributions, the acceptance probability $1/M'$ would be bounded by $1/M$. In our case, the acceptance probability can be bounded as follows (using Lemma 10):

$$\begin{aligned}
\frac{1}{M'} = \frac{|(1/N)\mathcal{R}^m \cap \mathcal{B}_{\mathcal{R},m}(r)|}{2|(1/N)\mathcal{R}^m \cap \mathcal{B}_{\mathcal{R},m}(r')|} &= \frac{|\mathcal{R}^m \cap \mathcal{B}_{\mathcal{R},m}(Nr)|}{2|\mathcal{R}^m \cap \mathcal{B}_{\mathcal{R},m}(Nr')|} \\
&\geq \frac{\mathsf{Vol}(\mathcal{B}_{\mathcal{R},m}(Nr - \sqrt{mn}/2))}{2\mathsf{Vol}(\mathcal{B}_{\mathcal{R},m}(Nr' + \sqrt{mn}/2))} \\
&= \frac{1}{2}\left(\frac{Nr - \sqrt{mn}/2}{Nr' + \sqrt{mn}/2}\right)^{mn}.
\end{aligned}$$

It now suffices to bound the latter term from below by $1/(cM) = 1/(2c(r'/r)^{mn})$. This inequality is equivalent to:

$$c \geq \frac{1}{2} \cdot \left(\frac{r}{r - \sqrt{mn}/(2N)}\right)^{mn} \cdot \left(\frac{r' + \sqrt{mn}/(2N)}{r'}\right)^{mn},$$

and to:

$$N \geq \frac{1}{c^{1/(mn)} - 1} \cdot \frac{\sqrt{mn}}{2}\left(\frac{c^{1/(mn)}}{r} + \frac{1}{r'}\right),$$

which allows to complete the proof. □

B.3 Proof of Lemma 7

Proof. Let $\mathbf{y} \in \mathcal{B}_{\mathcal{R},m}(Nr' + \sqrt{mn}/2)$ and set $\mathbf{z} = \lfloor \mathbf{y} \rceil$. Note that $\mathbf{z}$ is sampled (before the rejection step) with probability

$$\frac{\mathsf{Vol}(T_{\mathbf{z}} \cap \mathcal{B}_{\mathcal{R},m}(Nr' + \sqrt{mn}/2))}{\mathsf{Vol}(\mathcal{B}_{\mathcal{R},m}(Nr'))},$$

where $T_{\mathbf{z}}$ is the hypercube of $\mathcal{R}_{\mathbb{R}}^m$ centered in $\mathbf{z}$ with side-length 1. By the triangle inequality, this probability is equal to $1/\mathsf{Vol}(\mathcal{B}_{\mathcal{R},m}(Nr' + \sqrt{mn}/2)$ when $\mathbf{z} \in \mathcal{B}_{\mathcal{R},m}(Nr')$. Hence the distribution of the output is exactly $U(\mathcal{R}^m \cap \mathcal{B}_{\mathcal{R},m}(Nr'))$, as each element is sampled with equal probability and as the algorithm almost surely terminates (its runtime follows a geometric law of parameter the rejection probability).

It remains to consider the acceptance probability, which is:

$$\frac{\sum_{\mathbf{y} \in \mathcal{R}^m \cap \mathcal{B}_{\mathcal{R},m}(Nr')} \mathsf{Vol}(T_{\mathbf{y}} \cap \mathcal{B}_{\mathcal{R},m}(Nr' + \sqrt{mn}/2))}{\mathsf{Vol}(\mathcal{B}_{\mathcal{R},m}(Nr' + \sqrt{mn}/2))}.$$

By the triangle inequality and Lemma 10, it is

$$\frac{|\mathcal{R}^m \cap \mathcal{B}_{\mathcal{R},m}(Nr')|}{\mathsf{Vol}(\mathcal{B}_{\mathcal{R},m}(Nr' + \sqrt{mn}/2))} \geq \left(\frac{Nr' - \sqrt{mn}/2}{Nr' + \sqrt{mn}/2}\right)^{mn}.$$

Note that by our choice of N, this is $\geq 1/M_0$. □

C Fixed-Point Sampling

In this appendix, we explain how to sample from the discretized hyperball distribution using fixed-point arithmetic.

We first describe the representation of numbers and operations. A fixed-point number in precision p will consist in a p-bit signed integer $k \in \mathbb{Z} \cap [-2^{p-1}, 2^{p-1})$ along with an implicit scaling exponent e: the represented number is $x = k \cdot 2^{e-p} \in [-2^{e-1}, 2^{e-1})$. The data can for example be stored in a p-bit integer in two's complement representation. The scaling exponent e is not stored, it only exists on paper. For convenience, a precision p fixed-point number x with implicit exponent e will be referred to as a (p, e)-number.

When performing arithmetic operations on fixed-point numbers, particular care must be taken with overflows: in the analysis, we make sure that during the algorithm execution, any (p, e)-number x will satisfy $|x| < 2^{e-1}$. The following assumes no overflow occurs. We can add, subtract and negate (p, e)-numbers exactly (note that we only consider the situation where the operands of those operations share the same exponent). We assume that we can multiply (p, e_0)-number x_0 with a (p, e_1)-number x_1 into a $(p, e_\times)$-number $x_\times$ as if the multiplication was exact and then rounded to a nearest representable number. Finally, we assume that we can compute an inverse square-root of a (p, e)-number x into a (p, e')-number y with possibly slightly more error than that. This is summarized as follows:

$$x_0 \oplus x_1 = x_0 + x_1; \quad x_0 \ominus x_1 = x_0 - x_1; \quad \ominus x = -x;$$
$$|(x_0 \otimes^{e_\times}_{e_0,e_1} x_1) - (x_0 \cdot x_1)| \leq 2^{e_\times - p - 1}; \quad |(1/\sqrt{\cdot})^{e_t}_{e} - 1/\sqrt{x}| \leq 2^{e_t - p}.$$

For the sake of simplicity, we fix the precision p to 128 once and for all and never perform operations with numbers of different precisions.

C.1 Gaussian samples

Our hyperball-uniform sampler relies on an algorithm that samples from the continuous Gaussian distribution. In our fixed-point sampling, we will make do with fixed-point approximations to samples from the continuous Gaussian distribution. Instead of sampling from the continuous Gaussian distribution and rounding, we sample from the discrete Gaussian distribution. For the discrete Gaussian sampler, we can for example rely on [BBE+19].

Lemma 11. *Let $\sigma > 0$. Let $D_{\mathbb{Z},\sigma}$ (resp. D_σ) be the distribution D over $\mathbb{Z}$ (resp. $\mathbb{R}$) such that $D(k) \sim \exp(-k^2/(2\sigma^2))$ for all $k \in \mathbb{Z}$ (resp. $k \in \mathbb{R}$). Then we have:*

$$\Pr_{k \leftarrow D_{\mathbb{Z},\sigma}}[|k| \geq 14 \cdot \sigma] \leq 2^{-140} \quad \text{and} \quad \max_{|k| \leq 14 \cdot \sigma} \frac{D_{\mathbb{Z},\sigma}(k)}{\lfloor D_\sigma \rceil(k)} \leq \frac{1}{1 - 8/\sigma}.$$

Note that the statement could be rephrased using the smooth Rényi divergence introduced in [DFPS22].

Proof. Using the discrete Gaussian tail bound from [Lyu12, Lemma 4.4], the weight of $D_{\mathbb{Z},\sigma}$ out of the interval $[-14 \cdot \sigma, 14 \cdot \sigma]$ is $\leq 2^{-140}$. Using the Poisson Summation Formula, we have that:

$$\forall k \in \mathbb{Z},\ D_{\mathbb{Z},\sigma}(k) \leq \frac{\exp(-k^2/(2\sigma^2))}{\sigma\sqrt{2\pi}}.$$

Further, for $k \in \mathbb{Z} \cap [-14 \cdot \sigma, 14 \cdot \sigma]$, the following inequalities hold:

$$\begin{aligned}
\lfloor D_\sigma \rceil(k) &= \frac{1}{\sigma\sqrt{2\pi}} \int_{k-1/2}^{k+1/2} \exp(-x^2/(2\sigma^2))\mathrm{d}x \\
&\geq \frac{\exp(-k^2/(2\sigma^2))}{\sigma\sqrt{2\pi}} \cdot \exp(-(|k|+1/4)/(2\sigma^2)) \\
&\geq \frac{\exp(-k^2/(2\sigma^2))}{\sigma\sqrt{2\pi}} \cdot \big(1 - \frac{|k|+1/4}{2\sigma^2}\big) \\
&\geq \frac{\exp(-k^2/(2\sigma^2))}{\sigma\sqrt{2\pi}} \cdot \big(1 - \frac{8}{\sigma}\big).
\end{aligned}$$

This completes the proof. □

We will take $\sigma = 2^{124}$ and view the sample from $D_{\mathbb{Z},\sigma}$ as a $(128, 6)$-number obtained as the rounding of a perfect continuous Gaussian sample. Lemma 11 implies that a signature forger for the imperfect Gaussian sampler succeeds with essentially the same probability with the ideal Gaussian sampler.

C.2 From Gaussian samples to approximate hyperball-uniforms

In the following, we assume that we have access to arbitrarily many statistically independent (p, e)-numbers $\overline{y_i}$ that approximate (perfect) samples y_i from $D_1 = \mathcal{N}(0, 1)$. We first consider the algorithm of Figure 3 with radius 1. We apply it using such y_i's and fixed-point arithmetic, with appropriately chosen implicit exponents for each step. We show that the vector $\overline{\mathbf{y}}$ output by the approximate algorithm is close to the vector $\mathbf{y}$ output by the exact algorithm. As $\mathbf{y}$ is uniformly distributed in a hyperball, the computed vector $\overline{\mathbf{y}}$ is an approximation to such a sample.

We first bound the quantities involved during the computations. These bounds are for the exact quantities. To avoid overflows, we actually need them for the corresponding computed quantities. We will see later that as the numerical errors are low, the bounds still essentially hold. The bounds are probabilistic, and hold with probability extremely close to 1.

Lemma 12. *Let $d_{\min} = 6 \cdot 256 + 2$ and $d_{\max} = 11 \cdot 256 + 2$. The following bounds hold for all $d \in [d_{\min}, d_{\max}]$:*

$$\Pr_{y \leftarrow D_1}[|y| \geq 2^4] < 2^{-188},$$

$$\Pr_{\substack{y_i \leftarrow D_1 \\ \forall i \in [d]}}[\|\mathbf{y}\|^2 \geq 2^{12}] < 2^{-144}, \qquad \Pr_{\substack{y_i \leftarrow D_1 \\ \forall i \in [d]}}[\|\mathbf{y}\|^2 \leq 2^9] < 2^{-144}.$$

$$\Pr_{\mathbf{z} \leftarrow U(\mathcal{B}_{d-2}(1))}[|z_1| \geq 2^{-2}] < 2^{-150}.$$

Proof. The first probability is $1 - \mathrm{erf}(2^4/\sqrt{2})$. The two others can be bounded the Laurent-Massart bounds for the chi-squared distribution, i.e., for all d, t:

$$\Pr_{\substack{y_i \leftarrow D_1 \\ \forall i \in [d]}}[\|\mathbf{y}\|^2 \geq d + 2\sqrt{dt} + 2t] \leq \exp(-t),$$

$$\Pr_{\substack{y_i \leftarrow D_1 \\ \forall i \in [d]}}[\|\mathbf{y}\|^2 \leq d - 2\sqrt{dt}] \leq \exp(-t).$$

For the last bound, we use [DFPS22, Lemma A.13]. The probability is exactly $I_{1-1/\eta^2}((d+1)/2, 1/2)$ where I refers to the regularized incomplete Beta function and $1/\eta$ is probabilistic magnitude upper bound. The results follow from numerical computations. □

Throughout the execution of the approximate version of the algorithm of Figure 3, we fix the precision to $p \geq 64$. The implicit exponents vary depending on the algorithm step: the y_i's are represented by $(p,5)$-numbers, their squares by $(p,13)$-numbers, the squared-norm $\|\mathbf{y}\|^2$ by a $(p,13)$-number, the inverse-norm $1/\|\mathbf{y}\|$ by a $(p,-3)$-number and the output coordinates on $(p,-1)$-numbers.

Assume that we have $|\overline{y_i} - y_i| \leq \epsilon_0$ for all i, for some $\epsilon_0 \geq 2^{-p+5}/2 = 2^{-p+4}$. To avoid overflows of $\overline{y_i}$'s, it suffices that $|y_i| \leq 2^4 - 2^{-p+5} - \epsilon_0$. The first bound from Lemma 12 still holds for any $\epsilon_0 \leq 2^{-5}$.

We now consider the computations of the approximations $\overline{y_i^2}$'s to the y_i^2's. We have:

$$\begin{aligned}\left|\overline{y_i^2} - y_i^2\right| &\leq |(\overline{y_i} \otimes_{5,5}^{13} \overline{y_i}) - \overline{y_i}^2| + |\overline{y_i} - y_i| \cdot |\overline{y_i} + y_i| \\ &\leq 2^{-p+12} + |\overline{y_i} - y_i| \cdot [|\overline{y_i} - y_i| + 2|y_i|] \\ &\leq 2^{-p+12} + 2^6 \cdot \epsilon_0.\end{aligned}$$

As addition is exact, we obtain:

$$\left|\overline{\|\mathbf{y}\|^2} - \|\mathbf{y}\|^2\right| \leq d_{\max} \cdot (2^{-p+12} + 2^6 \cdot \epsilon_0) =: \epsilon_1.$$

To avoid overflow of $\overline{\|\mathbf{y}\|^2}$ and hence of the $\overline{y_i^2}$'s, it suffices that $\|\mathbf{y}\|^2 \leq 2^{12} - 2^{-p-13} - \epsilon_1$. The second bound from Lemma 12 still holds for any $\epsilon_0 \leq 2^{-5}$.

We continue with the inverse square root computation. The following holds:

$$\begin{aligned}\left|\overline{1/\|\mathbf{y}\|} - \frac{1}{\|\mathbf{y}\|}\right| &\leq \left|(1/\sqrt{\cdot})_{13}^{-3}(\overline{\|\mathbf{y}\|^2}) - \frac{1}{\overline{\|\mathbf{y}\|}}\right| + \left|\frac{1}{\|\mathbf{y}\|} - \frac{1}{\overline{\|\mathbf{y}\|}}\right| \\ &\leq 2^{-p-3} + \frac{|\overline{\|\mathbf{y}\|^2} - \|\mathbf{y}\|^2|}{2[\|\mathbf{y}\|^2 - |\overline{\|\mathbf{y}\|^2} - \|\mathbf{y}\|^2|]^{3/2}} \\ &\leq 2^{-p-3} + \frac{\epsilon_1}{2[2^9 - \epsilon_1]^{3/2}} \\ &\leq 2^{-p-3} + 2^{-15}(1 + 2^{-1})\epsilon_1 =: \epsilon_2,\end{aligned}$$

where the last inequality holds for any $\epsilon_0 \leq 2^{-15}$. To avoid overflow of $\overline{1/\|\mathbf{y}\|}$, it suffices that $1/\|\mathbf{y}\| \leq 2^{-4} - 2^{-p-3} - \epsilon_2$. The third bound from Lemma 12 still holds for any $\epsilon_0 \leq 2^{-15}$.

We finally evaluate the accuracy of the output vector $\overline{\mathbf{z}}$ with respect to $\mathbf{z} := (y_1, \ldots, y_d)^\top / \|\mathbf{y}\|^2$. We have, for all i:

$$\begin{aligned}\left|\overline{z_i} - z_i\right| &\leq \left|\overline{y_i} \otimes_{5,-3}^{-1} \overline{1/\|\mathbf{y}\|} - \overline{y_i} \cdot \overline{1/\|\mathbf{y}\|}\right| + \left|\overline{y_i} \cdot \overline{1/\|\mathbf{y}\|} - y_i/\|\mathbf{y}\|\right| \\ &\leq 2^{-p-2} + \left|\overline{1/\|\mathbf{y}\|} - 1/\|\mathbf{y}\|\right| \cdot |\overline{y_i}| + |\overline{y_i} - y_i|/\|\mathbf{y}\| \\ &\leq 2^{-p-2} + 2^5 \cdot \epsilon_2 + 2^{-4} \cdot \epsilon_0 =: \epsilon_3.\end{aligned}$$

To avoid overflow of $\overline{z_i}$, it suffices that $|z_i| \leq 2^{-2} - 2^{-p-3} - \epsilon_3$. The fourth bound from Lemma 12 still holds for any $\epsilon_0 \leq 2^{-20}$.

Note that ϵ_3 is of the order of $2^{14}2^{-p}$. This is a crude upper bound, as it assumes that errors are always in the same direction.

C.3 Using Approximate Hyperball-Uniforms

We consider the algorithm from Figure 5. Step 2 is performed exactly. For Step 1, we use a sample $\overline{\mathbf{z}}$ obtained as described in the previous subsection, and multiply it by a radius r'' that we assume to be given as a 64-bit fixed-point arithmetic number. Given $\overline{\mathbf{z}}$, this induces a change of the implicit exponent, and an additional tiny error term. As $\overline{z_i}$ belongs to $(-1/4, 1/4)$ and is within ϵ_4 from its corresponding z_i, we can prove that $\overline{r'' \cdot z_i}$ belongs to $(-r/4, r/4)$ and is within $r'' \cdot (\epsilon_4 + 2^{-63})$ from its corresponding $r'' \cdot z_i$.

Let $\mathbf{t}$ denote the rounded vector at Step 2. When rounded as in Step 2, both $\overline{r'' \cdot z_i}$ and $r'' \cdot z_i$ result in the same vector $\mathbf{t}$ when the distance from $r'' \cdot z_i$ to $\mathbb{Z} \cdot N$ is $< N/2 - (\epsilon_4 + 2^{-63})$. Let $D^{\text{ideal}}_{\text{contained}}$ be the distribution over $\mathbb{Z}^{mn} \cap \mathcal{B}(r')$ of the rounded vector $\mathbf{t}$ at Step 2, when the whole rounding hypercube is contained in the initial hyperball. Let $D^{\text{real}}_{\text{contained}}$ be the analogous distribution for the approximate version of the algorithm. From the discussion above, we have, for all $\mathbf{t} \in \mathbb{Z}^{mn}$:

$$\frac{D^{\text{real}}_{\text{contained}}(\mathbf{x})}{D^{\text{ideal}}_{\text{contained}}(\mathbf{x})} \geq \left(1 - \frac{2r''(\epsilon_4 + 2^{-63})}{N}\right)^{mn}.$$

Here we want to use [Pre17, Lemma 3]. We also need an upper bound counterpart to the above. For usability for up to 2^{67} samples via Rényi divergence arguments, it suffices that the relative error δ satisfies $\approx 2^{-37}$. In practice, we will be using $Nr' \in [2^{25}, 2^{28})$ as the sampling radius: if the sample is larger than that, we will not keep it.

C.4 Rejection Sampling with Approximate Distribution

In this subsection, we discuss what happens when we replace the ideal distribution used as a source for the rejection sampling by the real distribution.

Lemma 13. *Let $\mathbf{v}$ a vector, $M > 0$ and P^i, Q^i, Q^r be three probability distributions such that:*

$$R_\infty(Q^i \| Q^r) < +\infty \quad \textit{and} \quad R_\infty(Q^r \| Q^i) < +\infty \quad \textit{and} \quad R_\infty(P^i \| Q^i_{\pm \mathbf{v}}) \leq M.$$

Then if we use the bimodal rejection sampling strategy for Q^i and P^i with Q^r as a source, the resulting final distribution P^r is such that

$$R_\infty(P^r \| P^i) \leq R_\infty(Q^i \| Q^r) R_\infty(Q^r \| Q^i).$$

Proof. Let p^i and p^r denote the acceptance probability of a single step of rejection sampling in the ideal and real setup, respectively. As each is related to a single random variable following either Q^i or Q^r and then follows the same process, it holds that

$$\frac{p^i}{p^r} \leq R_\infty(Q^i \| Q^r).$$

Moreover, note that $p^i = 1/M$. Hence, we have

$$P^r : \mathbf{x} \mapsto \frac{Q^r(\mathbf{x} - \mathbf{v}) + Q^r(\mathbf{x} + \mathbf{v})}{Q^i(\mathbf{x} - \mathbf{v}) + Q^i(\mathbf{x} + \mathbf{v})} \cdot \frac{P^i(\mathbf{x})}{M \cdot p^r}.$$

The first fraction is the ratio of the probability of the event "Get a $\mathbf{y}$ such that $\mathbf{y} \pm \mathbf{v} = \mathbf{x}$" in the real (numerator) and ideal (denominator) setup. Hence, this ratio is bounded from above by $R_\infty(Q^r \| Q^i)$. Plugging the uppder bound for each fraction yields the result. □