HAETAE: Hyperball bimodAl modulE rejecTion signAture schemE[⋆]

Jung Hee Cheon[1,2], Hyeongmin Choe[1], Julien Devevey[3], Tim Güneysu[4], Dongyeon Hong[2], Markus Krausz[4], Georg Land[4], Junbum Shin[2], Damien Stehlé[3,5], and MinJune Yi[1,2]

[1] Seoul National University
{jhcheon, sixtail528, yiminjune}@snu.ac.kr
[2] Crypto Lab Inc.
{decenthong93, junbum.shin}@cryptolab.co.kr
[3] École Normale Supérieure de Lyon
{julien.devevey, damien.stehle}@ens-lyon.fr
[4] Ruhr Universität Bochum
{tim.gueneysu, Markus.Krausz, Georg.Land}@rub.de
[5] Institut Universitaire de France

Abstract. We present HAETAE, a new lattice-based signature scheme, that we are submitting to Korean Post-Quantum Cryptography Competition for Korean standards. While based on the Fiat-Shamir with Aborts paradigm like the NIST-selected Dilithium signature scheme, our design choices depart from it and aim for an improved complexity/compactness compromise. For the same security levels, our scheme has signature sizes 30% to 40% smaller and verification key sizes 20% smaller. (These figures are for the same number of repetitions as Dilithium.) Even if we rely on somewhat more complex operations, we expect our optimized version to run as fast as Dilithium. Moreover, most operations remain relatively simple, which should ease constant-time implementations and masking.

Keywords: Lattice Cryptography · Post-Quantum Cryptography · Digital Signatures.

1 Introduction

We introduce HAETAE, 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

[⋆] This work is submitted to the 'Korean Post-Quantum Cryptography Competition' (www.kpqc.or.kr).

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] introduced in [DDLL13] allows for the use of bimodal distributions in the signature generation. The signature is now of the form $\mathbf{y} + (-1)^b \mathbf{S}\mathbf{c}$, where $\mathbf{y}$ is sampled from a fixed distribution and $b \in \{0, 1\}$ is sampled uniformly. The 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. Our key generation algorithm just creates an MLWE sample $(\mathbf{A}_{\mathsf{gen}}, \mathbf{b} = \mathbf{A}_{\mathsf{gen}}\mathbf{s}_{\mathsf{gen}} + \mathbf{e}_{\mathsf{gen}})$ modulo q. The $k \times (k + \ell)$ matrix $\mathbf{A}$ is defined as $\mathbf{A} = (-2\mathbf{b} + q\mathbf{j} \mid 2\mathbf{A}_{\mathsf{gen}} \mid 2\mathbf{Id}_k) \bmod 2q$. It can be checked that the key-pair equation is indeed satisfied, for $\mathbf{s} = (1 \mid \mathbf{s}_{\mathsf{gen}} \mid \mathbf{e}_{\mathsf{gen}})$. The verification key consists of $(\mathbf{A}_{\mathsf{gen}}, \mathbf{b})$. As $\mathbf{A}_{\mathsf{gen}}$ 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 $\mathbf{b}$. 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 to modify 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 opposed to Dilithium, our decomposition of bits technique is a simple Euclidean division with centered remainder, for a well-chosen divisor. 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.

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 sampling of uniform integers modulo q.

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 3.2. 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 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 sizes are given in bytes. For the signature sizes, we first give the signature sizes currently implemented in the attached code, without rANS coding. We also give the expected signature sizes when using rANS coding, which we are working on.

Parameters sets	HAETAE120	HAETAE180	HAETAE260
Target security	120	180	260
q	64513	64513	64513
(k, ℓ)	(2,4)	(3,6)	(4,7)
η	1	1	1
τ	39	49	60
Classical hardness of forging	123 (100)	189 (156)	258 (216)
Quantum hardness of forging	108 (87)	166 (137)	227 (190)
Classical hardness of key-recovery	125	236	288
Quantum hardness of key-recovery	109	208	253
Signature size without rANS	3040	4064	5792
Expected Signature size with rANS	1473	2268	2737
Public key size	1056	1568	2080
Expected $\|\mathsf{sig}\| + \|\mathsf{vk}\|$	2529	3836	4817

Table 1: Security and sizes for our parameters sets

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% percent 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 decomposition of bits technique does not present any special case,
  - 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$^+$17] and Mitaka [EFG$^+$22]. 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 only relies on a (zero-centered) continuous Gaussian sampler, used to sample uniformly in hyperballs. The calls to it can also be massively parallelized. 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 simpler for HAETAE than in Falcon and Mitaka.

Limitations

- Our uniform distributions over hyperballs are over a discrete set. But to sample from it, we make use of computations over real numbers, which in practice are instantiated with floating-point numbers. Compared to Dilithium, this leads to implementation complications (approximations to real numbers) and analysis complications (computation of a sufficient precision to maintain security). As a work in progress, we are working on finding the best compromise between precision of the sampling, accuracy of the resulting distribution and efficiency.
- The key generation algorithm restarts if the largest singular value of the secret key is too large. While we chose our parameters such that this occurs only once on average, this is making the key generation algorithm slower than Dilithium's.

Comparison with Hash and Sign signatures. 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 lower case 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 $\mathbf{y}$).

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

Let $\mathcal{B}_{\mathcal{R},m}(r, \mathbf{c}) = \{\mathbf{x} \in \mathcal{R}_\mathbb{R}^m \mid \|\mathbf{x} - \mathbf{c}\| \leq r\}$ denote the (continuous) hyperball with radius $r > 0$ and center $\mathbf{c} \in \mathcal{R}^m$ 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 $r = r' \bmod \alpha$. We also define $r \bmod^+ \alpha$ as the unique integer r' in the range $[0, \alpha)$ satisfying $r = r' \bmod \alpha$. We naturally extend this to integer polynomials and vectors of integer polynomials, by applying it component-wise.

For a vector $\mathbf{y} = (y_1, \cdots, y_{nk})^\top \in \mathcal{R}_\mathbb{R}^k$, we define the matrix $\mathsf{rot}(\mathbf{y}) \in \mathbb{R}^{nk \times n}$ such that the $(ni + j, k)$ entry is the $(j - k)$-degree coefficient of y_{i-1} if $j - k \geq 0$, and the opposite of the $(n + j - k)$-degree coefficient for $0 \leq i < n$ and $j, k \in [1, n]$. We let the largest singular value of a real-valued matrix $\mathbf{S}$ be denoted by $\sigma_{\mathsf{max}}(\mathbf{S})$.

2.2 Lattice assumptions

We first recall the well-known lattice assumptions MLWE and MSIS on algebraic lattices, and the $\mathsf{SelfTargetMSIS}$ assumption from [KLS18].

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. HAETAE uses (discretized) uniform distributions over hyperballs, in the bimodal context. A 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

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

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

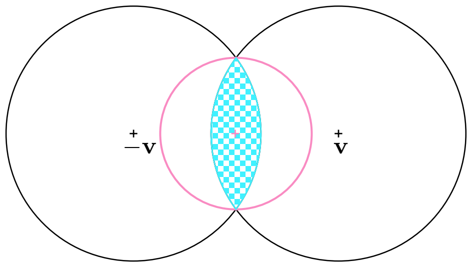

Fig. 1: The HAETAE eyes

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.

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

$\mathcal{B}$ :
  1: $\mathbf{z} \leftarrow U(\mathcal{B}_{(1/N)\mathcal{R},m}(r))$
  2: Return $\mathbf{z}$ with probability $1/M'$
  3: Else return $\perp$
```

Fig. 2: Bimodal hyperball rejection sampling

As we do not know the exact value of M', we cannot use algorithm $\mathcal{B}$ as a signature simulator in the security proof of HAETAE. Note that in the security proofs of lattice-based Fiat-Shamir with Aborts signatures, it is required to have an efficient simulator that simulates all iterations of the signature algorithm. Hence, simply replacing $\mathcal{B}$ with a version that always output $\mathbf{z}$ does not suffice. Our proposal is to use $\mathcal{A}(\mathbf{0})$ as efficient simulator: as $\mathbf{0}$ has norm at most t for any $t > 0$, algorithm $\mathcal{A}(\mathbf{0})$ has statistical distance 0 with $\mathcal{B}$ and thus with $\mathcal{A}(\mathbf{v})$ for any $\mathbf{v}$ with norm $\leq t$.

2.4 Sampling in a Hyperball

Lattice cryptography often relies on Gaussian distributions. As we depart from this choice, we explain how to sample uniformly on a hyperball, i.e., how we generate the sample $\mathbf{y}$ from Figure 2.

```
$\mathbf{y} \leftarrow U(\mathcal{B}_{(1/N)\mathcal{R},m}(r'))$
    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. 3: Discrete hyperball uniform sampling

Lemma 2. *Let n be the degree of $\mathcal{R}$, $M_0 \geq 1$, $r', m > 0$ and set*

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

At each iteration, the algorithm from Figure 3 succeeds with probability $\geq 1/M_0$. Moreover, the distribution of the output is $U(\mathcal{B}_{(1/N)\mathcal{R},m}(r'))$.

The proof of this lemma can also be found in Appendix B.

Now that we reduced the problem of uniformly sampling over a discretized hyperball to the case of the continuous hyperball, we explain how to do so. 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. 4: Hyperball uniform sampling

Lemma 3 ([VGS17]). *The distribution of the output of the algorithm in Figure 4 is $U(\mathcal{B}_{\mathcal{R},k}(r''))$.*

Using Lemma 2.4, we can conclude that if use the algorithms in Figures 1 to 4 and if we can sample from a normal distribution correctly, then the resulting distribution of $\mathbf{z}$ is indeed the uniform sample from the discretized hyperball. However, floating-point arithmetic used in normal distribution sampling and Steps 2 and 3 introduces numerical errors. We are currently working on analyzing these errors and on how large the precision should be to provably maintain security.

2.5 High, Low and Least Significant Bits

In our scheme, we compress part of the signature by sending only the so-called "high" bits of it. While our technique may be reminiscent of the one from Dilithium [DKL+18], it is somewhat simpler to implement, as no special case needs to be considered. We first recall the Euclidean division with centered remainder.

Lemma 4. *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.

This lets us define our decomposition.

Definition 3 (High, low and least significant bits). *Let $r \in \mathbb{Z}$. Let $r_2 = \lfloor (r+\alpha/2)/\alpha \rfloor$, $r' = r \bmod^{\pm} \alpha$, $r_0 = r' \bmod^{+} 2$, and $r_1 = (r'-r_0)/2$. We define:*

$$(\mathsf{LSB}(r), \mathsf{LowBits}(r,\alpha), \mathsf{HighBits}(r,\alpha)) \;=\; (r_0, r_1, r_2).$$

In HAETAE, we will LSB_{2q}, $\mathsf{LowBits}_{2q}$ and $\mathsf{HighBits}_{2q}$ defined over $\mathbb{Z}_{2q}$ as $\mathsf{LSB}_{2q}(r) = \mathsf{LSB}_{2q}(r \bmod^{+} 2q)$, $\mathsf{LowBits}_{2q}(r, \alpha) = \mathsf{LowBits}(r \bmod^{+} 2q, \alpha)$ and $\mathsf{HighBits}_{2q}(r, \alpha) = \mathsf{HighBits}(r \bmod^{+} 2q, \alpha)$. We extend these definitions to vectors by applying it component-wise. We state that this decomposition lets us recover the original element and bound the components of the decomposition.

Lemma 5. *Let $\alpha \geq 4$ a multiple of 4. Let $q > 2$ with $2q-1 < \lfloor (2q-1)/\alpha \rfloor \alpha + \alpha/2$ and $r \in [0, 2q)$. Then it holds that*

$$\begin{aligned}
&r = \alpha \cdot \mathsf{HighBits}(r, \alpha) + 2 \cdot \mathsf{LowBits}(r, \alpha) + \mathsf{LSB}(r),\\
&\mathsf{LowBits}(r, \alpha) \in [-\alpha/4, \alpha/4),\\
&\mathsf{HighBits}(r, \alpha) \in [0, \lfloor (2q-1)/\alpha \rfloor].
\end{aligned}$$

Finally, we also have

$$\mathsf{HighBits}(u\alpha + v, \alpha) = u$$

for any integers $u \geq 0$ and $v \in \{0, 1\}$.

Proof. Let $r' = r \bmod^{\pm} \alpha$. By definition, we have

$$r' = 2 \cdot \mathsf{LowBits}(r, \alpha) + \mathsf{LSB}(r).$$

By Lemma 4, there exists a unique representation

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

By definition of $\mathsf{HighBits}(r, \alpha)$, rewriting the above equations yields the first result.

By definition of $\bmod^{\pm}$, we have that $r' \in [-\alpha/2, \alpha/2)$. As $\alpha/2$ is even and $r_0 = r' \bmod^{+} 2$, we obtain that $r' - r_0 \in [-\alpha/2, \alpha/2)$. This leads to the claimed range for $\mathsf{LowBits}(r, \alpha)$.

For the second range, since $\lfloor (r + \alpha/2)/\alpha \rfloor$ is a non-decreasing function, it suffices 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.

Finally, for $u \geq 0$ and $v \in \{0, 1\}$, we have

$$\mathsf{HighBits}(u\alpha + v, \alpha) = u + \lfloor (\alpha/2 + v)/\alpha \rfloor = u,$$

since $0 < \alpha/2 + v < \alpha$. □

Note that $\mathsf{LSB}(r) = \mathsf{LSB}(r + 2y)$ for any r and y.

2.6 Signature Encoding via range Asymmetric Numeral System

In order to encode a signature, we will split it between its low and high bits. If we choose the number of low bits correctly, they will be distributed almost uniformly. We recall a specific type of entropy coding, named range Asymmetric Numeral systems (rANS) [Dud13]. When used to encode multiple coordinates that are identically distributed, rANS can be more efficient than Huffman coding applied on each coordinate.

Definition 4 (Range Asymmetric Numeral System (rANS) Coding). *Let $n > 0$ and $S \subseteq [0, 2^n - 1]$. Let $f : [0, 2^n - 1] \to \mathbb{Z} \cap (0, 2^n]$ such that $\sum_{x \in S} f(x) \leq 2^n$ and $f(x) = 0$ for all $x \notin S$. We define the following:*

- $\mathsf{CDF} : S \to \mathbb{Z}$, *defined as* $\mathsf{CDF}(s) = \sum_{y=0}^{s-1} f(y)$.
- $\mathsf{symbol} : \mathbb{Z} \to S$, *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*

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

Then, we define the rANS encoding/decoding for the set S and frequency $f/2^n$ as in Figure 5.

```
Encode((s_1, ..., s_m) ∈ S^m)
  1: $x_0 = 0$
  2: for $i = 0, \cdots, m-1$ do
  3:     $x_{i+1} = C(x_i, s_{i+1})$
  4: Return $x_m$

Decode($x \in \mathbb{Z}$)
  1: $y_0 = x$
  2: $i = 0$
  3: while $y_i > 0$ do
  4:     $t_{i+1} = \mathsf{symbol}(y_i \bmod^+ 2^n)$
  5:     $y_{i+1} = \lfloor y_i / 2^n \rfloor \cdot f(t_{i+1}) + (y_i \bmod^+ 2^n) - \mathsf{CDF}(t_{i+1})$
  6:     $i \leftarrow i + 1$
  7: $m = i - 1$
  8: Return $(t_m, \cdots, t_1) \in S^m$
```

Fig. 5: rANS encoding and decoding procedures

Lemma 6 (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*

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

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 apply this strategy to the high bits of the coordinates of a vector sampled following the rounding of a uniformly distributed over a discretized hyperball sample. We rely on a few heuristics. First, we choose to ignore the dependency between the coordinates, as we do not know how to handle it. We note that this dependency is likely to be limited, in the sense that it should heuristically become significant only when considering many coordinates. Second, we estimate the probability mass function p_s by replacing the discretized hyperball with the continuous one. Finally, we apply some rounding strategy to compute f such that the average overcost per coordinate caused by this rounding is almost negligible. We give the script used to compute the look-up tables as a supporting script of this submission. Please refer to the file "helper_script/compute_rans_table.py".

3 Specification

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 to get a first approach on HAETAE.

We give the high-level description of our signature scheme in Figure 6. In all of the following sections, we let $\mathbf{j} = (1, 0, \ldots, 0) \in \mathcal{R}^k$. The parameters ρ and α refer to the size of the seed and the compression factor, respectively. The parameter γ is the maximum allowed value for $\sigma_{\mathsf{max}}(\mathsf{rot}(\mathbf{s}))$, which ensures that $\|\mathbf{s}c\|_2 \leq \gamma \, \mathrm{wt}(c)$ for all $c \in \mathcal{R}_2$. The parameters B, B', and B'' refer to 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 $\mathbf{z} = (\mathbf{z}_1^\top, \mathbf{z}_2^\top)^\top$ with $\mathbf{z}_1 \in \mathcal{R}_\mathbb{R}^\ell$ and $\mathbf{z}_2 \in \mathcal{R}_\mathbb{R}^k$. At Step 3 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 satisfy the assumptions from Lemma 5.

```
KeyGen(1^λ)
  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: $\mathbf{A} = (-2\mathbf{b} + q\mathbf{j} \mid 2\mathbf{A}_{\mathsf{gen}} \mid 2\mathbf{Id}_k)$ and write $\mathbf{A} = (\mathbf{A}_1 \mid 2\mathbf{Id}_k)$
  4: $\mathbf{s} = (1, \mathbf{s}_{\mathsf{gen}}^\top, \mathbf{e}_{\mathsf{gen}}^\top)^\top$
  5: if $\sigma_{\mathsf{max}}(\mathsf{rot}(\mathbf{s}_{\mathsf{gen}})) > \gamma$, then restart
  6: Return $\mathsf{sk} = \mathbf{s}$, $\mathsf{vk} = \mathbf{A}$

Sign(sk, M)
  1: $\mathbf{y} \leftarrow U(\mathcal{B}_{(1/N)\mathcal{R},(k+\ell)}(B))$
  2: $c = H(\mathsf{HighBits}_{2q}(\mathbf{A}\lfloor\mathbf{y}\rceil, \alpha), \mathsf{LSB}_{2q}(\lfloor y_0\rceil \cdot \mathbf{j}), M) \in \mathcal{R}_2$
  3: $\mathbf{z} = (\mathbf{z}_1^\top, \mathbf{z}_2^\top)^\top = \mathbf{y} + (-1)^b c \cdot \mathbf{s}$ for $b \leftarrow U(\{0,1\})$
  4: $\mathbf{h} = \mathsf{HighBits}_{2q}(\mathbf{A}\lfloor\mathbf{z}\rceil - qc\mathbf{j}, \alpha) - \mathsf{HighBits}_{2q}(\mathbf{A}_1\lfloor\mathbf{z}_1\rceil - qc\mathbf{j}, \alpha) \in \mathcal{R}^k$
  5: if $\|\mathbf{z}\|_2 \geq B'$, then restart
  6: if $\|2\mathbf{z} - \mathbf{y}\|_2 < B$, then restart with probability 1/2
  7: Return $\sigma = (\mathsf{Encode}(\mathsf{HighBits}(\lfloor\mathbf{z}_1\rceil, a)), \mathsf{LowBits}(\lfloor\mathbf{z}_1\rceil, a), \mathsf{LSB}(\lfloor\mathbf{z}_1\rceil), \mathbf{h}, c)$

Verify(vk, M, σ = (x, v, w, h, c))
  1: $\tilde{\mathbf{z}}_1 = \mathsf{Decode}(x) \cdot 2^a + 2\mathbf{v} + \mathbf{w}$
  2: $\tilde{\mathbf{z}}_2 = (\mathbf{h} \cdot \alpha - 2 \cdot \mathsf{LowBits}_{2q}(\mathbf{A}_1\tilde{\mathbf{z}}_1 - qc\mathbf{j}, \alpha))/2 \bmod q$
  3: $\tilde{\mathbf{z}} = (\tilde{\mathbf{z}}_1^\top | \tilde{\mathbf{z}}_2^\top)^\top$
  4: $\mathbf{h}' = \mathsf{LSB}_{2q}((\tilde{z}_0 - c)\mathbf{j})$
  5: Return $\big( c = H(\mathsf{HighBits}_{2q}(\mathbf{A}\tilde{\mathbf{z}} - qc\mathbf{j}, \alpha), \mathbf{h}', M) \big) \wedge ( \|\tilde{\mathbf{z}}\| < B'' )$
```

Fig. 6: High-level description of HAETAE

3.1 Specification of HAETAE

We now give the full 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 $c \in \mathcal{R}_2$ with Hamming weight $\leq \tau$,
- Extendable output function expandA for deriving $\mathbf{A}_{\mathsf{gen}}$ from $\mathsf{seed}_{\mathbf{A}}$,
- Extendable output function expandS for deriving $\mathbf{s}$ from $\mathsf{seed}_{\mathsf{sk}}$, such that $\sigma_{\mathsf{max}}(\mathsf{rot}(\mathbf{s})) \leq \gamma$,
- Extendable output function expandYbb for deriving $\mathbf{y}$, b and b' from seed_{ybb} and counter,

The above building blocks can be implemented with symmetric primitives. For the $\mathsf{expandS}_\gamma$ function, however, there should be rejection sampling on the sampled secret to bound the size of the signature. If we bound the largest singular value of $\mathbf{s}$ as $\sigma_{\max}(\mathsf{Rot}(\mathbf{s})) \leq \gamma$, then we can bound $\|c\mathbf{s}\|_2 \leq \gamma \cdot \tau$. The secret will be sampled and tested whether the inequality holds. If it fails then restart (using extra pseudo-randomness). Note that γ is set to be the median of $\sigma_{\max}(\mathsf{Rot}(\mathbf{s}))$ for randomly chosen $\mathbf{s} \in S_\eta^{k+\ell}$ with always 1 in the first component.

Note that at Step 3 of the Verify algorithm, the division by 2 is well-defined as the operand is even and defined modulo $2q$.

We also give a fully randomized version of our scheme in Figure 8. It uses only expandA and H_{gen} among the above building blocks. We observe that in the randomized version signing process, the sampling algorithms for $\mathbf{y}$ and b can be performed "off-line", i.e., before receiving a message M to be signed. The same holds for other computations such as $\mathsf{HighBits}_{2q}(\mathbf{A}\lfloor\mathbf{y}\rceil, \alpha)$ and $\mathsf{LSB}_{2q}(\lfloor y_0\rceil \cdot \mathbf{j})$. In the "on-line" phase of signing, we can use $\mathbf{y}$ and b by choosing them randomly among the pre-sampled samples.

Lemma 7. *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) $\mathsf{HighBits}_{2q}(\mathbf{A}\tilde{\mathbf{z}} - qc\mathbf{j}, \alpha) = \mathsf{HighBits}_{2q}(\mathbf{A}\lfloor\mathbf{z}\rceil - qc\mathbf{j}, \alpha)$,
2) $\lfloor\mathbf{z}_2\rceil - \tilde{\mathbf{z}}_2 = \mathsf{LowBits}_{2q}(\mathbf{A}\tilde{\mathbf{z}} - qc\mathbf{j}, \alpha)$ *if* $B' + \alpha/4 \leq B'' < q/2$,
3) $\mathbf{h}' = \mathsf{LSB}_{2q}(\lfloor y_0\rceil \cdot \mathbf{j})$.

Proof. Let us prove the first statement. Let $\mathbf{w} = \mathbf{A}_1\lfloor\mathbf{z}_1\rceil - qc\mathbf{j}$. We have, modulo $2q$:

$$\begin{aligned}\mathbf{A}\tilde{\mathbf{z}} - qc\mathbf{j} &= \mathbf{w} + 2\tilde{\mathbf{z}}_2 \\ &= \mathbf{w} + \mathbf{h} \cdot \alpha - 2 \cdot \mathsf{LowBits}_{2q}(\mathbf{w}, \alpha) \\ &= \mathsf{LSB}_{2q}(\mathbf{w}) + \mathsf{HighBits}_{2q}(\mathbf{A}\lfloor\mathbf{z}\rceil - qc\mathbf{j}, \alpha) \cdot \alpha,\end{aligned}$$

where we used the definition of $\mathbf{h}$. Hence, it stems from Lemma 5 that,

$$\mathsf{HighBits}_{2q}(\mathbf{A}\tilde{\mathbf{z}} - qc\mathbf{j}, \alpha) = \mathsf{HighBits}_{2q}(\mathbf{A}\lfloor\mathbf{z}\rceil - qc\mathbf{j}, \alpha).$$

```
KeyGen($1^\lambda$)
 1: $\mathsf{seed} \leftarrow \{0,1\}^\rho$
 2: $(\mathsf{seed}_{\mathbf{A}}, \mathsf{seed}_{\mathsf{sk}}, K) = H_{\mathsf{gen}}(\mathsf{seed})$
 3: $\mathbf{A}_{\mathsf{gen}} \in \mathcal{R}_q^{k\times(\ell-1)} := \mathsf{expandA}(\mathsf{seed}_{\mathbf{A}})$
 4: $\mathbf{s} = (1, \mathbf{s}_{\mathsf{gen}}, \mathbf{e}_{\mathsf{gen}}) \in 1 \times S_\eta^{\ell-1} \times S_\eta^k := \mathsf{expandS}_\gamma(\mathsf{seed}_{\mathsf{sk}})$   // $\mathbf{s} \in S_\eta^{k+\ell}$
 5: $\mathbf{b} = \mathbf{A}_{\mathsf{gen}} \cdot \mathbf{s}_{\mathsf{gen}} + \mathbf{e}_{\mathsf{gen}} \bmod q$   // $\mathbf{b} \in \mathcal{R}_q^k$
 6: $\mathbf{A} = (\mathbf{A}_1 | \ 2\ \mathbf{Id}_k) = (-2\mathbf{b} + q\mathbf{j} | \ 2\mathbf{A}_{\mathsf{gen}} | \ 2\mathbf{Id}_k) \bmod 2q$   // $\mathbf{A} \in \mathcal{R}_{2q}^{k\times(k+\ell)}$
 7: Return $\mathsf{sk} = (\mathbf{A}, K, \mathbf{s})$, $\mathsf{vk} = (\mathsf{seed}_{\mathbf{A}}, \mathbf{b})$

Sign(sk, $M$)
 1: $\mu = H_{\mathsf{gen}}(\mathsf{seed}_{\mathbf{A}}, \mathbf{b}, M)$
 2: $\mathsf{seed}_{ybb} = H_{\mathsf{gen}}(K, \mu)$
 3: $\sigma = \perp$, $\mathsf{counter} = 0$
 4: while $\sigma = \perp$ do
 5:    $(\mathbf{y}, b, b') \in (\mathcal{B}_{(1/N)\mathcal{R},(k+\ell)}(B)) \times \{0,1\}^2 := \mathsf{expandYbb}(\mathsf{seed}_{ybb}, \mathsf{counter})$
 6:    $c = H(\mathsf{HighBits}_{2q}(\mathbf{A}\lfloor \mathbf{y} \rceil, \alpha), \mathsf{LSB}_{2q}(\lfloor y_0 \rceil \cdot \mathbf{j}), \mu)$   // $c \in \mathcal{R}_2$
 7:    $\mathbf{z} = (\mathbf{z}_1^\top, \mathbf{z}_2^\top)^\top = \mathbf{y} + (-1)^b c \cdot \mathbf{s}$
 8:    $\mathbf{h} = \mathsf{HighBits}_{2q}(\mathbf{A}\lfloor \mathbf{z} \rceil - qc\mathbf{j}, \alpha) - \mathsf{HighBits}_{2q}(\mathbf{A}_1\lfloor \mathbf{z}_1 \rceil - qc\mathbf{j}, \alpha)$   // $\mathbf{h} \in \mathcal{R}^k$
 9:    if $\|\mathbf{z}\|_2 \geq B'$, then $\sigma = \perp$
10:    if $\|2\mathbf{z} - \mathbf{y}\|_2 < B$ and $b' = 0$, then $\sigma = \perp$
11:    else $\sigma = (\mathsf{Encode}(\mathsf{HighBits}(\lfloor \mathbf{z}_1 \rceil, a)), \mathsf{LowBits}(\lfloor \mathbf{z}_1 \rceil, a), \mathsf{LSB}(\lfloor \mathbf{z}_1 \rceil), \mathbf{h}, c)$
12:    counter++
13: Return $\sigma$

Verify(vk, $M, \sigma = (x, \mathbf{v}, \mathbf{w}, \mathbf{h}, c)$)
 1: $\tilde{\mathbf{z}}_1 \leftarrow \mathsf{Decode}(x) \cdot 2^a + 2\mathbf{v} + \mathbf{w}$
 2: $\mathbf{A}_1 = (-2\mathbf{b} + q\mathbf{j} \mid 2 \cdot \mathsf{expandA}(\mathsf{seed}_{\mathbf{A}})) \bmod 2q$
 3: $\tilde{\mathbf{z}}_2 = (\mathbf{h} \cdot \alpha - 2 \cdot \mathsf{LowBits}_{2q}(\mathbf{A}_1\tilde{\mathbf{z}}_1 - qc\mathbf{j}, \alpha))/2 \bmod^{\pm} q$
 4: $\tilde{\mathbf{z}} = (\tilde{\mathbf{z}}_1^\top | \tilde{\mathbf{z}}_2^\top)^\top$
 5: $\mathbf{h}' = \mathsf{LSB}_{2q}((\tilde{z}_0 - c)\mathbf{j})$
 6: Return $\big(c = H(\mathsf{HighBits}_{2q}(\mathbf{A}\tilde{\mathbf{z}} - qc\mathbf{j}, \alpha), \mathbf{h}', H_{\mathsf{gen}}(\mathsf{seed}_{\mathbf{A}}, \mathbf{b}, M))\big) \wedge (\|\tilde{\mathbf{z}}\| < B'')$
```

Fig. 7: Deterministic version of HAETAE

We move on to the second statement. By successively using the form of $\mathbf{A}$ and the definitions of $\tilde{\mathbf{z}}_2$ and $\mathbf{h}$, and Lemma 5, we obtain, modulo $2q$:

$$\begin{aligned}
2\lfloor \mathbf{z}_2 \rceil - 2\tilde{\mathbf{z}}_2 &= \mathbf{A}\lfloor \mathbf{z} \rceil - qc\mathbf{j} - (\mathbf{w} + 2\tilde{\mathbf{z}}_2) \\
&\equiv \mathbf{A}\lfloor \mathbf{z} \rceil - qc\mathbf{j} - (\mathsf{HighBits}_{2q}(\mathbf{w}, \alpha) \cdot \alpha + \mathsf{LSB}_{2q}(\mathbf{w}) + \mathbf{h} \cdot \alpha) \\
&\equiv \mathbf{A}\lfloor \mathbf{z} \rceil - qc\mathbf{j} - \mathsf{HighBits}_{2q}(\mathbf{A}\lfloor \mathbf{z} \rceil - qc\mathbf{j}, \alpha) \cdot \alpha - \mathsf{LSB}_{2q}(\mathbf{w}) \\
&\equiv 2 \cdot \mathsf{LowBits}_{2q}(\mathbf{A}\lfloor \mathbf{z} \rceil - qc\mathbf{j}, \alpha) + \mathsf{LSB}_{2q}(\mathbf{A}\lfloor \mathbf{z} \rceil - qc\mathbf{j}) \\
&\quad - \mathsf{LSB}_{2q}(\mathbf{A}\lfloor \mathbf{z} \rceil - qc\mathbf{j}) \\
&\equiv 2 \cdot \mathsf{LowBits}_{2q}(\mathbf{A}\lfloor \mathbf{z} \rceil - qc\mathbf{j}, \alpha).
\end{aligned}$$

For the penultimate inequality, we used the fact that $\mathbf{w} = \mathbf{A}\lfloor \mathbf{z} \rceil - qc\mathbf{j} \bmod 2$. We then have $\tilde{\mathbf{z}}_2 \equiv \lfloor \mathbf{z}_2 \rceil - \mathsf{LowBits}_{2q}(\mathbf{A}\tilde{\mathbf{z}} - qc\mathbf{j}, \alpha) \bmod q$. We now show that this

```
$\underline{\mathsf{KeyGen}(1^\lambda)}$
 1: $\mathsf{seed} \leftarrow \{0,1\}^\rho$
 2: $\mathbf{A}_{\mathsf{gen}} = H_{\mathsf{gen}}(\mathsf{seed})$                                  // $\mathbf{A}_{\mathsf{gen}} \in \mathcal{R}_q^{k\times(\ell-1)}$
 3: $\mathbf{s} = \perp$
 4: while $\sigma_{\mathsf{max}}(\mathsf{rot}(\mathbf{s})) > \gamma$ do
 5:     $\mathbf{s} = (1, \mathbf{s}_{\mathsf{gen}}, \mathbf{e}_{\mathsf{gen}}) \leftarrow 1 \times S_\eta^{\ell-1} \times S_\eta^k$                 // $\mathbf{s} \in S_\eta^{k+\ell}$
 6: $\mathbf{b} = \mathbf{A}_{\mathsf{gen}} \cdot \mathbf{s}_{\mathsf{gen}} + \mathbf{e}_{\mathsf{gen}} \bmod q$                        // $\mathbf{b} \in \mathcal{R}_q^k$
 7: $\mathbf{A} = (\mathbf{A}_1 \mid 2\mathbf{Id}_k) = (-2\mathbf{b} + q\mathbf{j} \mid 2\mathbf{A}_{\mathsf{gen}} \mid 2\mathbf{Id}_k) \bmod 2q$   // $\mathbf{A} \in \mathcal{R}_{2q}^{k\times(k+\ell)}$
 8: Return $\mathsf{sk} = (\mathbf{A}, \mathbf{s})$, $\mathsf{vk} = (\mathsf{seed}, \mathbf{b})$

$\underline{\mathsf{Sign}(\mathsf{sk}, M)}$
 1: $\sigma = \perp$
 2: while $\sigma = \perp$ do
 3:     $\mathbf{y} \leftarrow U(\mathcal{B}_{(1/N)\mathcal{R},(k+\ell)}(B))$                         // $\mathbf{y} \in (1/N)\mathcal{R}^{k+\ell}$
 4:     $c = H(\mathsf{HighBits}_{2q}(\mathbf{A}\lfloor \mathbf{y} \rceil, \alpha), \mathsf{LSB}_{2q}(\lfloor y_0 \rceil \cdot \mathbf{j}), M)$       // $c \in \mathcal{R}_2$
 5:     $b \leftarrow \mathcal{U}(\{0,1\})$
 6:     $\mathbf{z} = (\mathbf{z}_1^\top, \mathbf{z}_2^\top)^\top = \mathbf{y} + (-1)^b c \cdot \mathbf{s}$
 7:     $\mathbf{h} = \mathsf{HighBits}_{2q}(\mathbf{A}\lfloor \mathbf{z} \rceil - qc\mathbf{j}, \alpha) - \mathsf{HighBits}_{2q}(\mathbf{A}_1\lfloor \mathbf{z}_1 \rceil - qc\mathbf{j}, \alpha)$   // $\mathbf{h} \in \mathcal{R}^k$
 8:     if $\|\mathbf{z}\|_2 \geq B'$, then $\sigma = \perp$
 9:     if $\|2\mathbf{z} - \mathbf{y}\|_2 < B$, then $\sigma = \perp$ with probability $1/2$
10:     else $\sigma = (\mathsf{Encode}(\mathsf{HighBits}(\lfloor \mathbf{z}_1 \rceil, a)), \mathsf{LowBits}(\lfloor \mathbf{z}_1 \rceil, a), \mathsf{LSB}(\lfloor \mathbf{z}_1 \rceil), \mathbf{h}, c)$
11: Return $\sigma$

$\underline{\mathsf{Verify}(\mathsf{vk}, M, \sigma = (x, \mathbf{v}, \mathbf{w}, \mathbf{h}, c))}$
 1: $\tilde{\mathbf{z}}_1 \leftarrow \mathsf{Decode}(x) \cdot 2^a + 2\mathbf{v} + \mathbf{w}$
 2: $\mathbf{A}_1 = (-2\mathbf{b} + q\mathbf{j} \mid 2 \cdot H_{\mathsf{gen}}(\mathsf{seed})) \bmod 2q$
 3: $\tilde{\mathbf{z}}_2 = (\mathbf{h} \cdot \alpha - 2 \cdot \mathsf{LowBits}_{2q}(\mathbf{A}_1\tilde{\mathbf{z}}_1 - qc\mathbf{j}, \alpha))/2 \mod^\pm q$
 4: $\tilde{\mathbf{z}} = (\tilde{\mathbf{z}}_1^\top | \tilde{\mathbf{z}}_2^\top)^\top$
 5: $\mathbf{h}' = \mathsf{LSB}_{2q}((\tilde{z}_0 - c)\mathbf{j})$
 6: Return $\big(\ c = H(\mathsf{HighBits}_{2q}(\mathbf{A}\tilde{\mathbf{z}} - qc\mathbf{j}, \alpha), \mathbf{h}', M)\ \big) \wedge (\ \|\tilde{\mathbf{z}}\| < B''\ )$
```

Fig. 8: Randomized version of HAETAE

equality actually holds over the integers. Note first that by definition of $\bmod^{\pm}$, we have $\tilde{\mathbf{z}}_2 \in [-q/2, q/2)^{nk}$. Further, by using the triangle inequality and Lemma 5, we have:

$$\|\lfloor \mathbf{z}_2 \rceil - \mathsf{LowBits}_{2q}(\mathbf{A}\tilde{\mathbf{z}} - qc\mathbf{j}, \alpha)\|_\infty \leq B' + \alpha/4 \leq B'' < q/2,$$

which implies the desired equality.

For the last statement, by considering only the first component of $\mathbf{z} = \mathbf{y} + (-1)^b c \cdot \mathbf{s}$, we obtain, modulo 2:

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

Multiplying by $\mathbf{j}$ yields the result. □

Theorem 1 (Completeness). *Assume that $B'' = B' + \sqrt{n(k+\ell)}/2 + \sqrt{nk} \cdot \alpha/4 < q/2$. 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:*

$$\mathsf{Verify}(\mathsf{vk}, M, \mathsf{Sign}(\mathsf{sk}, M)) = 1.$$

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. Then the first and last equations from Lemma 7 state that we will indeed have

$$c = H(\mathsf{HighBits}_{2q}(\mathbf{A}\tilde{\mathbf{z}} - qc\mathbf{j}, \alpha), \mathbf{h}', \mu).$$

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

$$\begin{aligned}
\|\tilde{\mathbf{z}}\| &\le \|\mathbf{z}\| + \|\mathbf{z} - \lfloor \mathbf{z} \rceil\| + \|\lfloor \mathbf{z} \rceil - \tilde{\mathbf{z}}\| \\
&\le B' + \sqrt{n(k+\ell)} \cdot \|\mathbf{z} - \lfloor \mathbf{z} \rceil\|_\infty + \|\lfloor \mathbf{z}_2 \rceil - \tilde{\mathbf{z}}_2\| \\
&\le B' + \frac{\sqrt{n(k+\ell)}}{2} + \sqrt{nk} \cdot \|\mathsf{LowBits}_{2q}(\mathbf{A}\lfloor \mathbf{z} \rceil - qc\mathbf{j}, \alpha)\|_\infty \\
&\le B' + \frac{\sqrt{n(k+\ell)}}{2} + \frac{\alpha\sqrt{nk}}{4}.
\end{aligned}$$

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

3.2 Parameter sets

We instantiate the HAETAE signature scheme to reach the NIST PQC security levels 2, 3 and 5. The instantiations are set to be at least as secure as the corresponding parameter sets for Dilithium and Falcon. We use the *core-SVP* methodology introduced in [ADPS16], a conservative security estimation method in lattice cryptography (see Section 5.2 for more details). The names of the three parameter sets correspond to the core-SVP security figures: HAETAE120 , HAETAE180 and HAETAE260. The parameters are provided in Table 2.

The ring dimension n and the modulus q are set to 256 and $64,513$ across all parameter sets. Our choice of modulus q allows for efficient integer sampling over $\mathbb{Z}_q$. This constraint leads in an unexpected estimated 236 bits of LWE security for HAETAE180. However, if we decrease ℓ by 1 (this is the parameter that has the most impact on the LWE security), we obtain only 175 bits of core-SVP hardness, which is below the 180 target.

Note that increasing k mostly increases the SIS security. Increasing η increases the LWE security, while decreasing the SIS security as it makes the SIS bound larger and should only be changed for fine-tuning. All of our estimations are computed using a modified version of the Dilithium security script, that we also submit as part of our submission package.

The variable B'' denotes the verification bound, which is half of the SIS bound. It is set significantly smaller than q, to avoid potential attacks exploiting the q-vectors: vectors with coordinates that are multiples of q always belong to the lattice corresponding to the cryptanalysis, and could potentially be used to improve lattice reduction attacks.

The figures between parentheses are for the strong unforgeability security in the case of the randomized signing version of HAETAE (in the deterministic version, strong and weak unforgeability are the same).

Parameters sets Target security	HAETAE120 120	HAETAE180 180	HAETAE260 260
n	256	256	256
q	64513	64513	64513
(k, ℓ)	(2,4)	(3,6)	(4,7)
η	1	1	1
τ	39	49	60
S	293.51	385	457.01
B	9779.3329	15709.1546	20614.9815
B'	9774.9271	15704.4361	20609.9152
B''	11197.4229	17973.1740	23740.4482
α	256	324	384
a	8	8	9
N	6161	7045	7254
BKZ block-size b to break SIS	421 (342)	647 (536)	885 (740)
Core-SVP classical hardness	123 (100)	189 (156)	258 (216)
Core-SVP quantum hardness	108 (87)	166 (137)	227 (190)
BKZ block-size b to break LWE	428	810	988
Core-SVP classical hardness	125	236	288
Core-SVP quantum hardness	109	208	253
Signature size without rANS	3040	4064	5792
Expected signature size with rANS	1473	2268	2737
Public key size	1056	1568	2080
Expected $\lvert\mathsf{sig}\rvert + \lvert\mathsf{vk}\rvert$	2529	3836	4817

Table 2: Parameter choices for 120, 180, 260 bits of core-SVP hardness

4 Performance analysis

In this section, we report the performance of our C reference implementation.

4.1 Description of platform

Tables 3 and 4 report the performance results of the reference implementation and the sizes. All benchmarks were obtained on one core of an AMD Ryzen 3700x processor clocked at 3589 MHz (as reported by `/proc/cpuinfo`) with TurboBoost and hyperthreading disabled. The benchmarking machine has 64 GB of RAM and is running Debian GNU/Linux with Linux kernel version 5.4.0. Implementations were compiled with gcc version 9.4.0 and the compiler flags as indicated in the CMakeLists included in the submission package.

4.2 Performance of reference implementation

All cycle counts reported are the median of the cycle counts of 1,000 executions of the respective function.

Scheme	KeyGen	Sign	Verify
HAETAE120	3094776	16088256	4571820
HAETAE180	6940764	30830184	9163440
HAETAE260	10777032	44705664	13739508

Table 3: Cycle counts for all parameter sets of HAETAE. Cycle counts were obtained on one core of an AMD Ryzen 7 3700X. For each funtion, this includes cycles of packing to bytes array from each data structure such as `sign key`, `verification key`, and `signature` and randombytes sampling.

We recall that the verification key consists of a seed of $\mathbf{A}_{\mathsf{gem}}$ and a polynomial vector b in $\mathcal{R}_q^k$. The signature consists of encoding of high bits of a polynomial vector $\lfloor \mathbf{z}_1 \rceil$, its low bits and LSB, a hint vector $\mathbf{h}$, and a challenge c. However, the current implementation outputs $\lfloor \mathbf{z}_1 \rceil$, $\mathbf{h}$ and c, i.e., without rANS encoding. Table 4 shows the currently implemented sizes of the verification key and signature in bytes.

Note that the representation of signatures is still a work in progress. Hence, as neither rANS or Huffman coding, the signatures are much bigger than what we could theoretically get. Our preliminary implementations of the rANS encoding, confirm that the final signature size will be very close to the expected size presented in Table 2. As this is just a matter of representation, it does not affect on security. We will provide the implementation with full compressions, in nearly future release.

Scheme	vk	σ
HAETAE120	1056	3040
HAETAE180	1568	4064
HAETAE260	2080	5792

Table 4: Key and signature sizes in bytes. The sizes reflect the current state of the implementation, yet without rANS coding.

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 that it can 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 then 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 [AFLT16] and the QROM [KLS18, GHHM21]. 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 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 first 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 5 **(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*

$$\mathsf{Adv}^{\mathsf{BimodalSelfTargetMSIS}}_{H,n,q,k,\ell,\beta}(\mathcal{A}) =$$

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

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, withprobability 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 $(-\mathbf{b} \mid \mathbf{A}_0 \mid \mathbf{Id}_k) \cdot (\mathbf{y} - \mathbf{y}') = 0 \bmod q$ and $\|\mathbf{y} - \mathbf{y}'\| < \beta$, it provides a $\mathsf{MSIS}_{n,q,k,\ell,\beta}$ solution for the matrix $(-\mathbf{b} \mid \mathbf{A}_0 \mid \mathbf{Id}_k)$, where the submatrix $(-\mathbf{b} \mid \mathbf{A}_0) \in \mathcal{R}_q^{k\times\ell}$ 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 aforementioned security notions for digital signatures.

Definition 6 (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:*

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

Definition 7 (Unforgeability under Chosen Message Attacks (UF-CMA)). *Let $\mathcal{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme. A UF-CMA adversary $\mathcal{A}$ has access to the verification key and a signing oracle, to which it can 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:*

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

Definition 8 (Strong Unforgeability under Chosen Message Attacks (SUF-CMA)). *Let $\mathcal{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme. An SUF-CMA adversary $\mathcal{A}$ has access to the verification key and a signing oracle, to which it can 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:*

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

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

Theorem 3 (UF-CMA Security of HAETAE in the QROM). *Let $B' \geq S/\sqrt{(M/2)^{2/m} - 1}$ and $B = (M/2)^{1/m} \cdot B'$ for $M > 1$ and $m = n(k+\ell)$. Assume that $\gamma \cdot \sqrt{\tau} \leq S$. Assume further that H is modeled as random oracle that the adversary can quantumly query (i.e., we place ourselves in the QROM). Then for any (quantum) UF-CMA adversary $\mathcal{A}$ of randomized HAETAE in Figure 8, there exist (quantum) adversaries $\mathcal{A}'$ and $\mathcal{A}''$ with similar run-times for MLWE and BimodalSelfTargetMSIS satisfying*

$$\mathsf{Adv}_{\mathsf{HAETAE}}^{\mathsf{UF\text{-}CMA}}(\mathcal{A}) \ \leq\ \mathsf{Adv}_{n,q,k,\ell-1,\eta}^{\mathsf{MLWE}}(\mathcal{A}') + \mathsf{Adv}_{H',n,q,k,\ell,B''}^{\mathsf{BimodalSelfTargetMSIS}}(\mathcal{A}'') + \varepsilon,$$

where ε is the min-entropy of the underlying identification protocol and H' is a random oracle derived from H.

Proof *(sketch).* Assume that there exists a (quantum) UF-CMA adversary $\mathcal{A}$ that can forge the signature with probability $\mathsf{Adv}^{\mathsf{UF\text{-}CMA}}(\mathcal{A})$. The identification protocol underlying HAETAE is zero-knowledge, meaning that it does not leak the secret, by Lemma 1. We recall from [KLS18] that there exists a (quantum) UF-NMA adversary $\mathcal{A}'$ that can forge the signature, i.e., without the signing oracle query.

Assuming $\mathsf{MLWE}_{n,q,k,\ell-1,\eta}$ is hard, adversary $\mathcal{A}'$ cannot distinguish the verification key component $\mathbf{b} = \mathbf{A}_{\mathsf{gen}}\mathbf{s}_{\mathsf{gen}} + \mathbf{e}_{\mathsf{gen}}$ from a truly uniform $\mathbf{b} \leftarrow \mathcal{R}_q^k$. Now, assume that adversary $\mathcal{A}'$ outputs a forgery which can be decoded to $\sigma = (\mathbf{z}_1, \mathbf{h}, c)$

for a message M satisfying $\mathsf{Verify}\,(\mathsf{vk}, M,\sigma) = 1$. For $\mathbf{u} = \mathsf{HighBits}_{2q}(\mathbf{A}\tilde{\mathbf{z}} - qc\mathbf{j}, \alpha)$, $\mathbf{h}'$ and $\tilde{\mathbf{z}}_2$ as in the verification algorithm, we have

$$\alpha \cdot \mathbf{u} + \mathbf{h}' \equiv \mathbf{A}_1\mathbf{z}_1 + 2\tilde{\mathbf{z}}_2 - qc\mathbf{j} \bmod 2q.$$

By the definition of $\mathsf{HighBits}$, $0 \le \|\alpha \cdot \mathbf{u} + \mathbf{h}'\|_\infty \le \left\lfloor \frac{2q-1}{\alpha} \right\rfloor$ holds, which implies that $\alpha \cdot \mathbf{u} + \mathbf{h}' = (\mathbf{A}_1\mathbf{z}_1 + 2\tilde{\mathbf{z}}_2 - qc\mathbf{j} \mod 2q)$. So we can rewrite the verification condition $c = H(\mathbf{u}, \mathbf{h}', M)$ as

$$\begin{aligned} c &= H\left(\frac{(\mathbf{A}_1\mathbf{z}_1 + 2\tilde{\mathbf{z}}_2 - qc\mathbf{j} \bmod 2q) - \mathbf{h}'}{\alpha}, \mathbf{h}', M\right) \\ &= H'\left(\mathbf{A}_1\mathbf{z}_1 + 2\tilde{\mathbf{z}}_2 - qc\mathbf{j} \bmod 2q, M\right), \end{aligned}$$

where $H' : \{-1, 0, \alpha - 1, \alpha, \cdots, \left\lfloor \frac{2q-1}{\alpha} \right\rfloor \cdot \alpha - 1, \left\lfloor \frac{2q-1}{\alpha} \right\rfloor \cdot \alpha\}^{nk} \times \{0,1\}^* \to \mathcal{R}_2$ is defined as:

$$H'(\alpha \cdot \mathfrak{u} - \mathfrak{h}, M) = H(\mathfrak{u}, \mathfrak{h}, M),$$

for any bit-string M and any integer vectors $\mathfrak{u}$ and $\mathfrak{h}$, whose components are in $[0, \left\lfloor \frac{2q-1}{\alpha} \right\rfloor]$ and $[0, 1]$, respectively. Note that there is a one-to-one correspondence between the inputs of H and H' as $\mathfrak{h} = (\mathfrak{a} \mod 2), \mathfrak{u} = (\mathfrak{a} - \mathfrak{h})/\alpha$ for $\mathfrak{a} = \alpha \cdot \mathfrak{u} + \mathfrak{h}$. The function H' can hence be modeled as a random oracle. We have $H'\left(\mathbf{A} \cdot (\mathbf{z}_1^\top, \tilde{\mathbf{z}}_2^\top)^\top \bmod 2q, M\right) = c$, with $\|(\mathbf{z}_1^\top, \tilde{\mathbf{z}}_2^\top)^\top\|_2 < B''$. This implies that we can construct an adversary $\mathcal{A}'$ that outputs a solution $((\mathbf{z}_1^\top, \tilde{\mathbf{z}}_2^\top)^\top, c, M)$ of $\mathsf{BimodalSelfTargetMSIS}_{H',n,q,k,\ell,B''}$ with advantage that is at least

$$\mathsf{Adv}^{\mathsf{UF\text{-}NMA}}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{MLWE}}_{n,q,k,\ell-1,\eta}(\mathcal{A}'') - \varepsilon,$$

for any polynomial-time $\mathsf{MLWE}_{n,q,k,\ell-1,\eta}$ adversary $\mathcal{A}''$. □

We now adapt the theorem statement for $\mathsf{SUF\text{-}CMA}$ security reduction, without proof.

Theorem 4 ($\mathsf{SUF\text{-}CMA}$ Security of HAETAE in the QROM). *Let the same conditions hold as in Theorem 3. Assume further that H_{gen}, $\mathsf{expandS}$ and $\mathsf{expandYbb}$ are pseudo-random, and $\mathsf{expandA}$ produces uniform sample in $\mathcal{R}_q^{k\times(k+\ell)}$. The adversary can quantumly query H and classically query the signing oracle. Then for any poly-time (quantum) $\mathsf{SUF\text{-}CMA}$ adversary $\mathcal{A}$ of the deterministic HAETAE in Fig. 7, there exist (quantum) adversaries $\mathcal{A}'$, $\mathcal{A}''$ and $\mathcal{A}'''$ for the MLWE, $\mathsf{BimodalSelfTargetMSIS}$ and MSIS such that*

$$\begin{aligned} \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}(\mathcal{A}) \le \mathsf{Adv}^{\mathsf{MLWE}}_{n,q,k,\ell-1,\eta}(\mathcal{A}') + \mathsf{Adv}^{\mathsf{BimodalSelfTargetMSIS}}_{H',n,q,k,\ell,B''}(\mathcal{A}'') \\ + \mathsf{Adv}^{\mathsf{MSIS}}_{n,q,k,\ell,2B''}(\mathcal{A}''') + \varepsilon. \end{aligned}$$

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 uses a Shortest Vector Problem (SVP) solver repeatedly 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 also determines the quality of the resulting basis. It provides a quality/time trade-off: If b gets larger, a better quality will be guaranteed but also the time complexity for the SVP solver will be exponentially increased. The time complexity of the b-BKZ algorithm is same with 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 is using the sieving method proposed by Becker et al. [BDGL16] which takes time $\approx 2^{0.292b+o(b)}$. The fastest known quantum variant was recently 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 the 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 5. 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 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, 3 and 4. 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.

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 $\Lambda_m = \{\mathbf{v} \in \mathbb{Z}^{\ell+m+1} : \mathbf{B}\mathbf{v} = \mathbf{0} \mod q\}$ for all $m \leq k$, where $\mathbf{B} = \left(\mathbf{A}_{[m]} \mid \mathbf{Id}_m \mid \mathbf{b}_{[m]}\right) \in \mathbb{Z}_q^{m\times(\ell+m+1)}$, $\mathbf{A}_{[m]}$ is the uppermost $m \times \ell$ submatrix 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 $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$. This implies that $(\mathbf{s} \mid \mathbf{e} \mid -1)$ is a short vector of Λ_m. The primal

Parameter sets	HAETAE120	HAETAE180	HAETAE260
Target security	120	180	260
BKZ block-size b to break SIS	421 (342)	647 (536)	885 (741)
Classical hardness	123 (100)	189 (156)	258 (216)
Quantum hardness	108 (87)	166 (137)	227 (190)
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 5: Core-SVP hardness for the best known attacks

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 structure of the input will not help finding 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 $(2\mathbf{b} \mid \mathbf{A}_0 \mid \mathbf{Id}_k) \cdot \mathbf{y}' = \mathbf{t} \mod q$, 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 in a more conservative way, 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 $(\mathbf{A} \mid \mathbf{Id}_k)$. 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]. To avoid such weaknesses, we choose the prime q larger than the MSIS bound β.

6 Conclusion

HAETAE follows the Fiat-Shamir with Abort design for lattice-based signatures. Our goal was to keep as much as possible from the conceptual simplicity of Dilithium and in particular avoid complex rejection conditions, while pushing the scheme in a more efficient direction regarding signature size. Overall, this leads to a simplicity/compactness compromise between Dilithium and Falcon, both standardized in the NIST PQC process.

HAETAE is a scheme with shorter signatures than Dilithium and that is easier to implement and protect against side-channel attacks than Falcon or Mitaka. We believe that the advantages we gain from this compromise outbalance the extra complications in signing we introduced. Indeed, while our scheme is not as compact as Falcon or Mitaka, our gain of 30% to 40% on signature size compared to Dilithium justifies having a scheme that is somewhat harder to implement.

Future works and directions. We believe that our signature scheme provides an interesting in-between between the two selected candidates of the NIST post-quantum cryptography projects. However, our distribution choice is unusual in lattice-based cryptography. This means in particular that we need to develop specific tools to handle this change. We give here a few directions that we are currently exploring in order to improve HAETAE.

Theoretical work. First, we are looking into the sampling algorithm for $\mathbf{y}$. As of now, the algorithm description (see Figure 4) considers exact computations over the reals, but the implementation relies on floating-point arithmetic. We intend to make this sampling rigorous and have the description and implementation coincide. First, we will derive a sufficient precision for the sampling computations, by analyzing the growth of numerical errors. We do not envision particular difficulties here, as similar techniques have been used for [FHK+17] (based on [Pre17]). Second, we will consider replacing floating-point arithmetic by fixed-point arithmetic. This should also work as these differ only when numbers of widely different magnitudes are being considered: thanks to Gaussian tail bounds, very large real numbers have vanishingly small probabilities of occurring; and vector coordinates that would be very close to 0 have a negligible impact on the rounding of the algorithm from Figure 3.

We are also studying how to best implement arithmetic $\bmod 2q$, notably using the Chinese remainder theorem (using a $\bmod 2$ part and a $\bmod q$ part).

Finally, while our choice of distribution is new, we note that the cryptanalyses approach we considered also apply to Dilithium, and have hence been well-studied already. It is however different when one considers leakage, as the signing algorithms differ significantly. We would like to better understand this aspect.

Implementation work. In the key generation algorithm, we did not implement yet the rejection condition on the maximum singular value of a matrix derived from the secret key. Also, we will add the rANS coding described in the present documentation in the code. Further, when any of the above questions will have been handled, we will adapt the implementation accordingly.

We are also considering side-channel resistance of the scheme. We intend to make the implementation constant-time. We will also consider a masked implementation, in particular once floating-point arithmetic will have been removed.

Acknowledgments. The authors thank to Jai Hyun Park and Wonhee Cho for helpful discussions. Julien Devevey and Damien Stehlé were supported by the AMIRAL ANR grant (ANR-21-ASTR-0016), the PEPR quantique France 2030 programme (ANR-22-PETQ-0008) and the PEPR Cyber France 2030 programme (ANR-22-PECY-0003).

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`.

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.

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.

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. Cryptology ePrint Archive, Number 2022/1249, 2022. `https://eprint.iacr.org/2022/1249`.

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`.

EFG$^+$22. 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.

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.

GHHM21. Alex B. Grilo, Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. Tight adaptive reprogramming in the QROM. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT*, pages 637–667. Springer, 2021.

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.

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.

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 $\mathbf{z}$.

```
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: $\mathbf{A} = (-2\mathbf{b} + q\mathbf{j} \mid 2\mathbf{A}_{\mathsf{gen}} \mid 2\ \mathbf{Id}_k)$ and write $\mathbf{A} = (\mathbf{A}_1 \mid 2\ \mathbf{Id}_k)$
  4: $\mathbf{s} = (1, \mathbf{s}_{\mathsf{gen}}^\top, \mathbf{e}_{\mathsf{gen}}^\top)^\top$
  5: Return $\mathsf{sk} = \mathbf{s}$ and $\mathsf{vk} = \mathbf{A}$

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

Verify(vk, $M, \sigma = (\tilde{\mathbf{z}}, c)$)
  1: Return ( $c = H(\mathbf{A}\tilde{\mathbf{z}} - qc\mathbf{j}, M)$ ) $\wedge$ ( $\|\tilde{\mathbf{z}}\| < B''$ )
```

Fig. 9: High-level description of uncompressed HAETAE

B Discretizing Hyperballs

B.1 Useful Lemma

We will rely on the following claim.

Lemma 8. *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 $r' \ge \sqrt{r^2 + t^2}$.

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 8):

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

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

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

and to:

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

which allows to complete the proof. □

B.3 Proof of Lemma 2

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 8, 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 $\geq 1/M_0$. ◻