

ATLAS: A Compact Module-LWR Signature Scheme*

Karthick Srivatsan¹, Debranjana Pal², Anindya Ganguly¹, Suparna Kundu³, Abhinava De⁴, Puja Mondal¹, Harry Hart⁵, Quinten Norga³, Prajna Mahadev⁴, Supriya Adhikary¹, Debayan Das⁴, Chaoyun Li⁵, and Angshuman Karmakar¹

¹ Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, India
{karthicks24, anindya, pujamondal, adhikarys, angshuman}@cse.iitk.ac.in

² The LNM Institute of Information Technology, India
debranjana.crl@gmail.com

³ COSIC, KU Leuven, Kasteelpark Arenberg 10, Bus 2452, B-3001 Leuven-Heverlee, Belgium
{suparna.kundu, quinten.norga}@esat.kuleuven.be,

⁴ Indian Institute of Science, Bangalore, India
{abhinavade, debayandas, prajnam24}@iisc.ac.in

⁵ University of Surrey, United Kingdom
{h.hart, c.li}@surrey.ac.uk

Abstract. We present ATLAS, a lattice-based digital signature scheme built on the Fiat–Shamir with aborts paradigm, with security based on the hardness of the Module Learning with Rounding (MLWR) problem. Unlike Dilithium’s $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ construction or HAETAE’s bimodal, hyperball-uniform instantiation, ATLAS derives its public key via deterministic rounding, $\mathbf{t} = \lfloor \frac{p}{q}\mathbf{A}\mathbf{s}_1 \rfloor$, eliminating the error term $\mathbf{s}_2$ and the explicit noise sampling it requires. This reduces key-generation cost and secret-key storage, and removes a component that both Dilithium and HAETAE would otherwise need to support, while keeping ATLAS structurally close to Dilithium to inherit its well-studied design and cryptanalytic track record.

ATLAS targets three classical security levels, 128, 256, and 512 bits, using the smallest feasible ring degree n and challenge weight κ for each, with module dimensions (k, l) tuned per level. To our knowledge, this includes the first MLWR-based signature parameter set at the 512-bit level. ATLAS further adopts power-of-two moduli (e.g., $q = 2^{23}$, $p = 2^{18}$), replacing modular reduction with bitwise operations and removing rejection sampling from key parts of the scheme. Since such moduli preclude conventional NTT-based multiplication, we design a hierarchical Toom-Cook/Karatsuba decomposition with degree-8 schoolbook multiplication as its base case, accelerated via AVX2 vectorization and strided memory access. Finally, ATLAS supports both SHAKE and KDF-SM3 as interchangeable symmetric back-ends, and we quantify the overhead of the SM3-based instantiation relative to SHAKE.

Keywords: Post-quantum Cryptography · Lattices · LWR Problem.

1 Introduction

Lattice-based post-quantum signatures. The advent of large-scale quantum computers threatens the security of virtually all deployed public-key cryptography, as Shor’s algorithm [38] efficiently solves the integer factorization and discrete logarithm problems underlying RSA and elliptic-curve schemes. In response, lattice-based cryptography has emerged as one of the leading candidates for post-quantum security, combining strong worst-case-to-average-case hardness guarantees with practical efficiency [35]. This is reflected in the recent standardisation of CRYSTALS-Dilithium [21] by the National Institute of Standards and Technology [4], and of HAETAE [17] as a winning digital signature scheme of South Korea’s national post-quantum cryptography competition (KpqC), announced in January 2025 [24].

Motivation. While Dilithium and HAETAE demonstrate that the Fiat-Shamir with aborts paradigm yields practical lattice-based signatures, both schemes base their security on the Module Learning with Errors and Module Short Integer Solution (MLWE/MSIS) problems, which require sampling explicit noise during key generation and, for Dilithium, uniform rejection sampling during signing. From a hardware perspective, both remain a persistent source of implementation cost and side-channel risk: rejection sampling over non-power-of-two moduli, in particular, introduces variable-time branches that must be carefully hidden to achieve constant-time behavior, motivating research into simpler, hardware-friendly alternatives.

* ATLAS appeared in Round 1 of the Next-generation Commercial Cryptographic (NGCC) Algorithms Program.

The Learning with Rounding (LWR) problem, introduced by Banerjee *et al.* [7] as a derandomized variant of Learning with Errors (LWE) [36], offers such an alternative: noise is replaced by deterministic rounding from a modulus q down to a smaller modulus p , removing the need for noise sampling and, when q, p are powers of two, replacing modular reductions with bitwise shifts and masks. Okada *et al.* [33] first explored this with MLWRSign, to our knowledge the first LWR-based signature scheme, but it fixes a single ring degree ($n = 256$) and challenge weight across *all* security levels, leaving performance on the table at lower levels and offering no support for higher ones.

ATLAS addresses this limitation, bridging theoretical security and practical hardware constraints to target extreme parallelizability and a minimal memory footprint on resource-constrained IoT devices while scaling gracefully to high-throughput platforms. Its security rests on MLWR, whose deterministic rounding removes the need for pseudo-random generation of an explicit error vector, shrinking public-key size and improving resistance to fault-injection attacks that exploit seed or nonce manipulation during error sampling [34]. Unlike MLWRSign, ATLAS scales $n \in \{128, 256, 512\}$ with the target security level, following design-space exploration methodologies used for efficient KEMs [26, 28]. Smaller rings reduce hardware resource usage and memory footprint (e.g., on the ARM Cortex-M4) and enable greater parallelism, while larger rings deliver maximum post-quantum margin. Throughout, ATLAS uses power-of-two moduli (e.g., $q = 2^{23}$, $p = 2^{18}$ for ATLAS-128), eliminating modular reduction and rejection sampling from the arithmetic path; since this precludes the NTT, ATLAS instead uses Toom–Cook and Karatsuba-based multiplication, matching NTT-based performance at the chosen degrees while gaining free modular reduction. Its Fiat-Shamir-with-aborts security bounds are chosen to resist state-of-the-art lattice reduction attacks and preserve the target security level (128, 256, or 512 bits) even under public-key truncation, with p and the truncation parameter further tuned to minimize key and signature size subject to these same margins.

Contribution. In this work, we introduce ATLAS, a lattice-based digital signature scheme built on the *Fiat-Shamir with aborts* framework whose security rests on the Module LWR (MLWR) problem. Our contributions are as follows.

- **An MLWR-based signature design without explicit noise sampling.** Unlike Dilithium’s MLWE relation $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ and HAETAE’s bimodal, hyperball-uniform MLWE instantiation, ATLAS generates its public key as $\mathbf{t} = \lfloor \frac{p}{q} \mathbf{A}\mathbf{s}_1 \rfloor$, where q is the working modulus, $p < q$ is the smaller rounding modulus, $\mathbf{A}$ is a public matrix over the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$ of ring degree n , and $\mathbf{s}_1$ is the secret key vector. This eliminates the error term $\mathbf{s}_2$ and the explicit noise sampling it requires, reducing key generation cost, shrinking secret key size, and removing an implementation component that both Dilithium and HAETAE must still support in some form.

- **Security-level-specific parameterization.** We propose parameters for 128-bit (ATLAS-128), 256-bit (ATLAS-256) and 512-bit (ATLAS-512) security levels, (see Table 3). In contrast to MLWRSign [33], which fixes the ring degree $n = 256$ and the challenge weight κ (the Hamming weight of the challenge polynomial c) to $\kappa = 60$ for every security level, ATLAS selects the smallest feasible n and κ that meet each target security level, like $n = 128$ and $\kappa = 31$ for ATLAS-128.

- **Power-of-two arithmetic for hardware efficiency.** ATLAS uses power-of-two moduli such as $q = 2^{23}$ and $p = 2^{18}$ (see Table 3), replacing complex modular reduction with bitwise operations and eliminating rejection sampling in key parts of the scheme. At the 128-bit security level this yields a public key of only 1328 bytes and a signature of 2081 bytes (see Table 4).

- **A hierarchical polynomial multiplication strategy.** Because power-of-two moduli preclude conventional NTT-based multiplication, we design and implement a decomposition framework combining Toom–Cook and Karatsuba algorithms with degree-8 schoolbook multiplication as the base case (Section 5). We further evaluate strided variants of each strategy and characterize when they improve execution time and memory utilization. Empirically, the best strategies are two levels of Toom-4 followed by degree-8 schoolbook multiplication for $n = 128$; two levels of Toom-4, one level of Karatsuba, and degree-8 schoolbook for $n = 256$; and three levels of Toom-4 followed by degree-8 schoolbook for $n = 512$.

- **ATLAS using SHAKE and SM3 hash functions.** Our ATLAS supports both SHAKE [31] and the KDF-SM3 framework [39], enabling deployment in both international standard and SM3-mandated environments without altering the underlying algebraic construction. We provide a detailed performance comparison of ATLAS instantiated with each hash function in Table 8.

2 Preliminaries

Notation and preliminaries Let $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$ and $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$, where n is a power-of-two ring degree and $q > p$ are power-of-two moduli. Elements of $\mathbb{Z}_q$ are identified with their unique representatives in $[-q/2, q/2) \cap \mathbb{Z}$. Bold

lower-case and upper-case letters denote vectors and matrices over $\mathcal{R}$ (or $\mathcal{R}_q$), respectively. For $\mathbf{a} \in \mathcal{R}^l$, $\|\mathbf{a}\|_\infty$ denotes the maximum absolute coefficient across all entries, and $\|\mathbf{a}\|_1$ denotes the sum of absolute coefficients. For an integer $b \geq 0$, we define $S_b = \{a \in \mathcal{R} : \|a\|_\infty \leq b\}$. The challenge space $\mathcal{B}_\kappa \subset \mathcal{R}$ consists of polynomials with exactly κ non-zero coefficients in $\{-1, 1\}$. For masking and decomposition parameters γ_1, γ_2 , we denote the scaled bounds as $\bar{\gamma}_i = (p/q)\gamma_i$. The parameter $d = pk_{\text{drop}}$ denotes the number of least significant bits dropped from public-key components, and ω bounds the maximum hint weight. The hash function CRH is a collision-resistant hash instantiated using either SHAKE [31] or SM3 [39], and H denotes the random oracle mapping commitments and messages into $\mathcal{B}_\kappa$.

2.1 Hard Problems

We present the hard problems, like Module Learning With Rounding (MLWR), Module Short Integer Solution (MSIS), and Self-Target MSIS (SelfTargetMSIS) problems. The security of our design relies on these mathematical hard problems.

Definition 1 (MLWR Problem). Let $\text{pp} = (k, l, S)$ be the problem parameters. Let $\mathcal{U}$ denote the uniform distribution over $\mathcal{R}_q^{k \times l} \times \mathcal{R}_p^k$. The MLWR_{pp} distribution samples $(\mathbf{A}, \mathbf{t} = \lfloor \frac{p}{q} \mathbf{A} \mathbf{s} \rfloor)$, where $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{k \times l}$ and $\mathbf{s} \xleftarrow{\$} S^l \subset \mathcal{R}_q^l$. The advantage of a distinguisher $\mathcal{D}$ for the decisional MLWR problem is:

$$\text{Adv}_{\text{MLWR}}^{\text{pp}}(\mathcal{D}) := \left| \Pr \left[\mathcal{D}(\mathbf{A}, \mathbf{t}) = 1 \mid (\mathbf{A}, \mathbf{t}) \xleftarrow{\$} \mathcal{U} \right] - \Pr \left[\mathcal{D}(\mathbf{A}, \mathbf{t}) = 1 \mid (\mathbf{A}, \lfloor \frac{p}{q} \mathbf{A} \mathbf{s} \rfloor) \xleftarrow{\$} \text{MLWR}_{\text{pp}} \right] \right|$$

Definition 2 (MSIS Problem). Given a problem parameters $\text{pp} = \{k, l, \gamma\}$, and a matrix $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{k \times l}$, the MSIS_{pp} problem requires finding a vector $\mathbf{v} \in \mathcal{R}^{k+l}$ such that $0 < \|\mathbf{v}\|_\infty \leq \gamma$ and $[\mathbf{I}_k \mid \mathbf{A}] \cdot \mathbf{v} = \mathbf{0}_k \in \mathcal{R}_q^k$. The advantage of an adversary $\mathcal{A}$ is:

$$\text{Adv}_{\text{MSIS}}^{\text{pp}}(\mathcal{A}) := \Pr \left[0 < \|\mathbf{v}\|_\infty \leq \gamma \wedge [\mathbf{I}_k \mid \mathbf{A}] \cdot \mathbf{v} = \mathbf{0}_k \mid \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{k \times l}, \mathbf{v} \leftarrow \mathcal{A}(\mathbf{A}) \right]$$

Definition 3 (SelfTargetMSIS Problem). Let $H : \{0, 1\}^* \rightarrow \mathcal{B}_\kappa$ be a cryptographic hash function, $\text{pp} = (H, k, l, \zeta)$. Given a matrix $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{k \times l}$, the $\text{SelfTargetMSIS}_{\text{pp}}$ problem requires finding a message μ and a vector $\mathbf{v}$ such that $0 < \|\mathbf{v}\|_\infty \leq \gamma$ and $H(\mu \parallel [\mathbf{I}_k \mid \mathbf{A}] \cdot \mathbf{v}) = c$. The advantage of an adversary $\mathcal{A}$ is:

$$\text{Adv}_{\text{SelfTargetMSIS}}^{\text{pp}}(\mathcal{A}) := \Pr \left[0 < \|\mathbf{v}\|_\infty \leq \gamma \wedge H(\mu \parallel [\mathbf{I}_k \mid \mathbf{A}] \cdot \mathbf{v}) = c \mid \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{k \times l}; (\mathbf{v}, \mu) \leftarrow \mathcal{A}^{H(\cdot)}(\mathbf{A}) \right]$$

2.2 Supporting Algorithms

The proposed signature scheme incorporates several auxiliary subroutines to manage the trade-off between precision and storage while ensuring correctness in the presence of rounding errors (see Figure 1). The bit decomposition functions `decompose_keygen`, and `decompose_sign` partition a ring element $a \in \mathbb{Z}_q$ into a high-order component a_1 and a low-order remainder a_0 . This mechanism facilitates the generation of a compact public key $pk := (\rho, \mathbf{t}_1)$ during `KeyGen` and enables the signer to derive the challenge c from $\mathbf{w}_1$ during `sig_sign`. To compensate for the loss of precision caused by public key truncation, the hinting algorithms `make_hint` and `use_hint` identify and resolve potential carry-overs. Specifically, the signer employs `make_hint` to detect if the secret-key-dependent term $\mathbf{cs}_2$ alters the high-order bits of the commitment. This allows the verifier to use `use_hint` to reconstruct $\mathbf{w}'_1$ and successfully validate the signature. Note that the output of each `decompose` function is categorized as (high bits, low bits).

3 Fundamentals of ATLAS

3.1 Basic Construction

Our design adopts the *Fiat-Shamir with Aborts* framework, employing a design methodology consistent with that of Dilithium [21]. The following discussion presents key generation, signature generation and the verification algorithm (see Figure 2, 3).

(a) decompose_keygen(a) 1. $a_1 := (a - 1 + 2^{pk_{drop}-1}) \gg pk_{drop}$ 2. $a_0 := q + a - (a_1 \ll pk_{drop})$ 3. return (a_1, a_0)	(b) decompose_sign(a) 1. $a_1 := (a - 1 + \bar{\gamma}_2) \gg \text{bits}_{\bar{\gamma}_1}$ 2. $a_0 := p + a - (a_1 \ll \text{bits}_{\bar{\gamma}_1})$ 3. $a_1 \leftarrow a_1 \& ((p \gg \text{bits}_{\bar{\gamma}_1}) - 1)$ 4. return (a_1, a_0)
(c) make_hint(a, b) 1. $v := (a + b) \& (p - 1)$ 2. $a_1 \leftarrow \text{decompose_sign}(a)$ 3. $a_2 \leftarrow \text{decompose_sign}(v)$ 4. if $a_1 \neq a_2$ then 5. return 1 6. else 7. return 0	(d) use_hint($a, hint$) 1. $(a_1, a_0) \leftarrow \text{decompose_sign}(a \& (p - 1))$ 2. if $hint = 0$ then return a_1 3. $mask := (p \gg \text{bits}_{\bar{\gamma}_1}) - 1$ 4. if $a_0 > p$ then 5. return $(a_1 + 1) \& mask$ 6. else 7. return $(a_1 - 1) \& mask$

Fig. 1: decompose_keygen, decompose_sign, make_hint, and use_hint procedures

Key Generation. $(pk, sk) \leftarrow \text{sig_keygen}(1^\lambda)$. The *key generation algorithm* starts by sampling a seed ρ and seed K , where the public matrix $\mathbf{A} \in \mathcal{R}_q^{k \times l}$ is deterministically expanded from the seed ρ using `expand_mat`. The secret component consists of a short vector $\mathbf{s}_1$ sampled uniformly from S_η^l . The algorithm then computes the public vector $\mathbf{t} \leftarrow \lfloor \frac{p}{q} \mathbf{A} \mathbf{s}_1 \rfloor \in \mathcal{R}_p^k$. To achieve a compact public key, $\mathbf{t}$ is partitioned into $\mathbf{t}_1$ and $\mathbf{t}_0$ using a decomposition (`decompose_keygen`) at a power-of-two boundary 2^d , and outputs $\mathbf{t}_1$ as a compressed version of $\mathbf{t}$ as public key. A collision-resistant hash function is used to compute $tr = \text{CRH}(\rho \parallel \mathbf{t}_1)$. The algorithm concludes by outputting the public key $pk = (\rho, \mathbf{t}_1)$ and the secret key $sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{t}_0)$.

sig_keygen() 1: $\rho \xleftarrow{\$} \{0, 1\}^{256}, K \xleftarrow{\$} \{0, 1\}^{256}$ 2: $\mathbf{A} \in \mathcal{R}_q^{k \times l} := \text{expand_mat}(\rho)$ and $\mathbf{s}_1 \xleftarrow{\$} S_\eta^l$ 3: $\mathbf{t} := \lfloor \frac{p}{q} \mathbf{A} \mathbf{s}_1 \rfloor \in \mathcal{R}_p^k$ 4: $(\mathbf{t}_1, \mathbf{t}_0) := \text{decompose_keygen}(\mathbf{t}, 2^d)$ and $tr := \text{CRH}(\rho \parallel \mathbf{t}_1)$ 5: $pk := (\rho, \mathbf{t}_1)$ and $sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{t}_0)$ 6: return pk, sk	sig_verify(pk, σ, M) 1: $\mathbf{A} \in \mathcal{R}_q^{k \times l} := \text{expand_mat}(\rho)$ 2: $\mu \leftarrow \text{CRH}(\text{CRH}(pk) \parallel M)$ 3: $\mathbf{w}'_1 \leftarrow \text{use_hint}(\mathbf{h}, \lfloor \frac{p}{q} \mathbf{A} \mathbf{z} \rfloor - c \mathbf{t}_1 \cdot 2^{pk_{drop}}, 2\bar{\gamma}_2)$ 4: $c' := \text{H}(\mu \parallel \mathbf{w}'_1)$ 5: if $\ \mathbf{z}\ _\infty < \gamma_1 - \beta_1$ and $c = c'$ and $\text{HW}(\mathbf{h}) \leq \omega$ then 6. return 1 7: else 8. return 0
--	--

Fig. 2: sig_keygen and sig_verify procedures

Signature Generation. $\sigma \leftarrow \text{sig_sign}(sk, M)$. The *signature algorithm* begins by reconstructing the public matrix $\mathbf{A}$ from the seed ρ and deriving the auxiliary secret component $\mathbf{s}_2 = \mathbf{t} - \frac{p}{q} \mathbf{A} \mathbf{s}_1$. A message-dependent digest μ is generated using the public key hash tr and the message M . The signer then enters an iterative loop where a masking vector $\mathbf{y}$ is sampled to produce a commitment $\mathbf{w} = \lfloor \frac{p}{q} \mathbf{A} \mathbf{y} \rfloor$, along with its associated rounding error ξ_1 . The challenge c is computed by hashing the high-order bits of the commitment, $\mathbf{w}_1$, together with μ . Following this, the potential response $\mathbf{z} = \mathbf{y} + c \mathbf{s}_1$ and a hint vector $\mathbf{h}$ are calculated. Here, $\mathbf{h}$ assists the verifier by compensating for rounding errors involving the public key component $\mathbf{t}_0$. To prevent information leakage and ensure verifiability, the algorithm performs two rejection sampling: at first, it verifies that the coefficients of $\mathbf{z}$ and $\mathbf{r}_0$ are within prescribed bounds, it means $\mathbf{z} \in S_{\gamma_1 - \beta_1 - 1}^l$, and $\mathbf{r}_0 \in S_{\bar{\gamma}_2 - \beta_2 - 1}^k$, and $\mathbf{r}_1 = \mathbf{w}_1$. It ensures that $\mathbf{z}$ does not leak any information about the secret vectors $\mathbf{s}_1, \mathbf{s}_2$. The second rejection sampling ensures that the Hamming weight (HW) of the hint vector $\mathbf{h}$ is less than ω and $\|\mathbf{t}_0\|_\infty < \bar{\gamma}_2$. This ensures verifiability as mentioned. If these rejection sampling conditions are not met, the algorithm increments the counter and restarts. Upon success, the final signature $\sigma = (\mathbf{z}, \mathbf{h}, c)$ is returned.

Verification Algorithm. $0/1 \leftarrow \text{sig_verify}(pk, \sigma, M)$. The *verification algorithm* determines the validity of a signature $\sigma = (\mathbf{z}, \mathbf{h}, c)$ for a message M . The verifier reconstructs the public matrix $\mathbf{A}$ and the message digest μ . Using the hint vector $\mathbf{h}$, the verifier reconstructs the high-order bits of the commitment, $\mathbf{w}'_1$, from the term $\lfloor \frac{p}{q} \mathbf{A} \mathbf{z} \rfloor - c \mathbf{t}_1 \cdot 2^{pk_{drop}}$. The signature is accepted if the challenge c matches the hash of the reconstructed bits and all norm constraints on $\mathbf{z}$ and $\mathbf{h}$ are satisfied.

```

sig_sign( $sk, M$ )
1:  $\mathbf{A} \in \mathcal{R}_q^{k \times l} := \text{expand\_mat}(\rho)$ 
2:  $\mathbf{t} = \mathbf{t}_1 \cdot 2^d + \mathbf{t}_0$ 
3:  $\mathbf{s}_2 := (\mathbf{t} - \frac{p}{q} \mathbf{A} \mathbf{s}_1)$ 
4:  $\mu := \text{CRH}(tr \parallel M)$ 
5:  $\text{count} := 0$ 
6: repeat
7:    $\text{count} \leftarrow \text{count} + 1$ 
8:    $\mathbf{y} \in S_{\gamma_1-1}^l := \text{Sam}(K \parallel \mu \parallel \text{count})$ 
9:    $\mathbf{w} := \lfloor \frac{p}{q} \mathbf{A} \mathbf{y} \rfloor \in \mathcal{R}_p^k$ 
10:   $\boldsymbol{\xi}_1 := \lfloor \frac{p}{q} \mathbf{A} \mathbf{y} \rfloor - \frac{p}{q} \mathbf{A} \mathbf{y}$ 
11:   $(\mathbf{w}_1, \_) := \text{decompose\_sign}(\mathbf{w}, 2\bar{\gamma}_2)$ 
12:   $c := H(\mu \parallel \mathbf{w}_1) \in \mathcal{B}_\kappa$ 
13:   $\mathbf{z} := \mathbf{y} + c \mathbf{s}_1$ 
14:   $\boldsymbol{\xi}_2 := \lfloor c \mathbf{s}_2 \rfloor - c \mathbf{s}_2$ 
15:   $\boldsymbol{\nu} := \lfloor \boldsymbol{\xi}_2 - \boldsymbol{\xi}_1 \rfloor$ 
16:   $(\mathbf{r}_1, \mathbf{r}_0) := \text{decompose\_sign}(\mathbf{w} - \lfloor c \mathbf{s}_2 \rfloor + \boldsymbol{\nu}, 2\bar{\gamma}_2)$ 
17:   $\mathbf{h} := \text{makehint}(-c \mathbf{t}_0, \mathbf{w} - \lfloor c \mathbf{s}_2 \rfloor + \boldsymbol{\nu} + c \mathbf{t}_0, 2\bar{\gamma}_2)$ 
18:   $\text{accept} \leftarrow \text{True}$ 
19:  if  $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta_1$  or  $\|\mathbf{r}_0\|_\infty \geq \bar{\gamma}_2 - \beta_2$  or  $\mathbf{r}_1 \neq \mathbf{w}_1$  then
20:     $\text{accept} \leftarrow \text{False}$ 
21:  if  $\text{HW}(\mathbf{h}) > \omega$  or  $\|c \mathbf{t}_0\|_\infty \geq \bar{\gamma}_2$  then
22:     $\text{accept} \leftarrow \text{False}$ 
23: until  $\text{accept} = \text{True}$ 
24:  $\sigma \leftarrow (\mathbf{z}, \mathbf{h}, c)$ 
25: return  $\sigma$ 

```

Fig. 3: sig_sign procedure

3.2 Properties of ATLAS

In this section, we explain the basic security properties for ATLAS. We evaluate the *correctness*, *unforgeability*, *beyond-unforgeability* properties.

1. Correctness A digital signature scheme is considered *correct* if the verification algorithm always accepts an honestly generated signature. We present a formal proof for the correctness of ATLAS.

Theorem 1 (Correctness). *The ATLAS = {sig_keygen, sig_sign, sig_verify} signature scheme is correct if, for any signature $\sigma = (\mathbf{z}, \mathbf{h}, c)$ generated by sig_sign(sk, M), the reconstructed high-order bits $\mathbf{w}'_1$ satisfy $\mathbf{w}'_1 = \mathbf{w}_1$. This equality is guaranteed if the rejection sampling criteria in sig_sign ensure that the error terms associated with $\mathbf{s}_1, \mathbf{s}_2$, and $\mathbf{t}_0$ remain within the correction range of the use_hint function.*

Proof (Proof Sketch). We rewrite the verification expression as $\mathbf{w} - \lfloor c \mathbf{s}_2 \rfloor + \boldsymbol{\nu} + c \mathbf{t}_0$. Because the accumulated rounding and secret noise falls strictly within the hint bound $\bar{\gamma}_2$, the hint recovery accurately reconstructs the high-order bits $\mathbf{w}'_1 = \mathbf{w}_1$. The full proof is detailed in Appendix B.

2. Unforgeability The standard security goal for digital signature schemes is *Existential Unforgeability under Chosen-Message Attack* (EUF-CMA). In this model, an adversary is provided with the public key and granted adaptive access to a signing oracle. The goal of the adversary is to produce a valid signature on a message that has not been previously queried. A more stringent requirement is *Strong Unforgeability* (SUF-CMA), which considers the adversary successful if it produces a different signature for a message that has already been signed by the oracle.

Evaluating the ATLAS scheme's security in a post-quantum context requires moving beyond the classical Random Oracle Model (ROM) into the Quantum Random Oracle Model (QROM). While classical reductions to Module Learning with Rounding (MLWR) and Module Short Integer Solution (MSIS) rely on the *Forking Lemma*, this technique is fundamentally

incompatible with quantum adversaries. Because a quantum forger may query the hash function in a superposition of states, the act of *rewinding* to extract a second signature is hindered by the no-cloning theorem and the collapse of the quantum state upon measurement. Consequently, proving **SUF-CMA** security for **ATLAS** in the **QROM** requires alternative reduction strategies that do not depend on simple rewinding.

The following Theorem 2 establishes the **SUF-CMA** security of **ATLAS** signature in the quantum random oracle model (**QROM**). The security of the construction relies mainly on the three lattice problems **MLWR**, **MSIS**, and **SelfTargetMSIS**. Basically, the **MLWR** problem ensures security against key recovery, the **SelfTargetMSIS** assumption provides security against new-message forgery, and the **MSIS** problem is required for strong unforgeability.

Theorem 2 (SUF-CMA Security of ATLAS). *Suppose $\mathcal{QA}$ is a quantum adversary against the **ATLAS** signature scheme that makes q_H queries to the quantum random oracle $|H\rangle$. Then there exist quantum adversaries $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ that solve the **MLWR** problem with parameters $\text{pp}_1 = \{k, l, S\}$, the **SelfTargetMSIS** assumption with parameters $\text{pp}_2 = \{H, k, l + 1, \zeta\}$, and the **MSIS** problem with parameters $\text{pp}_3 = \{k, l, \zeta'\}$, respectively, such that the following holds.*

$$\text{Adv}_{\text{ATLAS}}^{\text{SUF-CMA}}(\mathcal{QA}) \leq \text{Adv}_{\text{MLWR}}^{\text{pp}_1}(\mathcal{B}) + \text{Adv}_{\text{SelfTargetMSIS}}^{\text{pp}_2}(\mathcal{C}) + \text{Adv}_{\text{MSIS}}^{\text{pp}_3}(\mathcal{D}) + 2^{-\tau+1}$$

Here, S denote the uniform distribution of the secret in the **MLWR** problem and τ represent the min-entropy of the underlying identification scheme. The other notations, ζ and ζ' , are defined as follows:

$$\begin{aligned} \zeta &= \max\{\gamma_1 - \beta_1, \frac{q}{p}(2\tilde{\gamma}_2 + 1 + \kappa \cdot 2^{p^{k_{drop}-1}})\} \leq 4\gamma_2 \\ \zeta_1 &= \max\{2(\gamma_1 - \beta_1), 4\gamma_2 + 2\} \leq 4\gamma_2 + 6\eta \end{aligned}$$

The reduction from **SUF-CMA** to **EU-F-KOA** (Existential Unforgeability under Key-Only Attack) security follows the framework of Kiltz, Lyubashevsky, and Schaffner [23]. It proved that for *zero-knowledge deterministic* signature schemes, any adversary having quantum access to the random oracle $|H\rangle$ and classical access to a signing oracle that can forge a signature on a new message can be converted into an adversary that forges without any access to the signing oracle. The applicability of this result to **ATLAS** depends on two things. First, the signing algorithm is *deterministic* in the sense that the randomness $\mathbf{y}$ is derived via a pseudorandom function of the message and secret key, so the scheme fits the deterministic signing model of [23]. Second, the underlying interactive protocol of **ATLAS** signing algorithm satisfies *non abort honest verifier zero-knowledge*: by the rejection sampling argument, the output distribution of $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$ is statistically independent of the secret $\mathbf{s}_1$, and signatures can therefore be perfectly simulated without the signing key⁶. Together, these two properties guarantee that signing queries provide the adversary with no information beyond what is already available from the public key and the quantum random oracle, so the signing oracle is redundant. Applying the [23] result directly therefore yields

$$\text{Adv}_{\text{ATLAS}}^{\text{SUF-CMA}}(\mathcal{QA}) \leq \text{Adv}_{\text{ATLAS}}^{\text{EU-F-KOA}}(\mathcal{QA}'),$$

where $\mathcal{QA}'$ runs in essentially the same time as $\mathcal{QA}$.

Lemma 1 (EU-F-KOA Security). *Suppose $\mathcal{QA}$ is a quantum adversary against the **EU-F-KOA**-security of the **ATLAS** signature scheme that issues q_H queries to the quantum random oracle $|H\rangle$. Then, there exist quantum adversaries $\mathcal{QB}$ and $\mathcal{QC}$ that solve the **MLWR** problem with parameters $\text{pp}_1 = \{k, l, S\}$ and the **SelfTargetMSIS** problem with parameters $\text{pp}_2 = \{H, k, l + 1, \zeta\}$, respectively, such that the following holds.*

$$\text{Adv}_{\text{ATLAS}}^{\text{EU-F-KOA}}(\mathcal{QA}) \leq \text{Adv}_{\text{MLWR}}^{\text{pp}_1}(\mathcal{QB}) + \text{Adv}_{\text{SelfTargetMSIS}}^{\text{pp}_2}(\mathcal{QC})$$

Furthermore, the time complexity is bounded by $\text{Time}(\mathcal{QB}) = \text{Time}(\mathcal{QC}) = \text{Time}(\mathcal{QA}) + q_H$.

The complete proof of Lemma 1 is provided in Appendix C.

Security Beyond Unforgeability Beyond standard unforgeability, **ATLAS** satisfies the *Beyond UnForgeability Features* (**BUFF**) security properties introduced by Cremers *et al.* [18]. The **BUFF** framework captures three security properties

⁶ Here we sketch the reduction idea. For more details about *naHVZK* proof, we refer [23]

that go beyond EUF-CMA: *exclusive ownership* (M-S-UEO), which ensures that a signature verifies under only one public key and prevents key substitution attacks; *message-bound signatures* (MBS), which ensures that a signature is bound to a unique message; and *non-resignability* (NR), which ensures that an adversary cannot produce a valid signature under a fresh key for an unknown message. The absence of these properties has been shown to enable real-world protocol attacks, including certificate hijacking attacks on early versions of the ACME protocol and path integrity attacks on the DRKey/OPT routing protocol [18].

The ATLAS signature scheme achieves all three BUFF properties (M-S-UEO, MBS, and NR) via Theorem 4.5 of Cremers et al. [18]. This theorem requires two structural conditions on the signature scheme: first, the signature must contain an explicitly verified hash digest of the message and the public key; second, the underlying hash function must be collision resistant and Φ -non-malleable for the class $\Phi = \{\phi_{pk'} \mid \phi_{pk'}(pk, M) = (pk', M)\}$. We verify both conditions for ATLAS below.

Structural condition. The message digest is computed as $\mu = \text{CRH}(\text{CRH}(\rho \parallel \mathbf{t}_1) \parallel M)$, where $(\rho, \mathbf{t}_1)$ constitutes the public key and M is the message being signed (see Figure 3). The challenge polynomial is then derived as $c \leftarrow H(\mu \parallel \mathbf{w}_1)$, and the signature output is $\sigma = (\mathbf{z}, \mathbf{h}, c)$. The verification algorithm recomputes μ from the public key and message, recomputes $\mathbf{w}_1$ from $(\mathbf{z}, \mathbf{h})$, and accepts only if the recomputed challenge matches c . Therefore, c is an explicitly verified digest of both the public key $(\rho, \mathbf{t}_1)$ and the message M , and it is included in σ . This is precisely the BUFF structure of [18], and carries no additional size overhead since c is already a required component of the signature.

Conditions on the Hash Function. In ATLAS, symmetric operations are instantiated using a collision-resistant hash function CRH, supporting either SHAKE [31] or SM3 [39]. The collision resistance of CRH follows directly from standard assumptions on the sponge construction (for SHAKE) and the underlying compression function (for SM3). To establish Φ -non-malleability, we adopt the argument of Cremers et al. [18] for iterative and sponge-based hash functions. Under this framework, producing a valid challenge c' for an altered public key pk' on an unknown message M requires querying CRH on the prefix containing pk' and M , namely $(\text{CRH}(\rho' \parallel \mathbf{t}'_1), M)$. Because M has super-logarithmic min-entropy conditioned on the adversary's view, evaluating this query without knowing M is computationally infeasible, establishing Φ -non-malleability.

Achieving BUFF Security. Finally, both conditions of Theorem 4.5 of [18] are established, yielding the target guarantees directly. First, M-S-UEO follows from the collision resistance of CRH, as any signature σ valid under two distinct public keys $pk_1 \neq pk_2$ yields an immediate hash collision. Similarly, MBS holds because a single signature valid for two distinct messages $M_1 \neq M_2$ under the same public key breaks collision resistance. Finally, NR follows from Φ -non-malleability: an adversary given a valid signature $\sigma = (\mathbf{z}, \mathbf{h}, c)$ on an unknown message M cannot forge a signature σ' under a fresh key pk' without evaluating $\text{CRH}(\text{CRH}(\rho' \parallel \mathbf{t}'_1) \parallel M)$, which is infeasible without knowledge of M .

3.3 Rejection Sampling Analysis

We analyze the rejection-sampling conditions in the signing algorithm to ensure that the output signature $\mathbf{z}$ and the hint vector $\mathbf{h}$ do not leak information about the secret key. Specifically, a signing iteration in Figure 3 succeeds if and only if all of the following rejection conditions are simultaneously satisfied:

- Event₁: $\|\mathbf{z}\|_\infty < \gamma_1 - \beta_1$ (Line 19)
- Event₂: $\|\mathbf{r}_0\|_\infty < \bar{\gamma}_2 - \beta_2$ (Line 19)
- Event₃: $\|[\mathbf{cs}_2] - \boldsymbol{\nu}\|_\infty \leq \beta_2$, which ensures $\mathbf{r}_1 = \mathbf{w}_1$ (Line 19)
- Event₄: $\|\mathbf{ct}_0\|_\infty < \bar{\gamma}_2$ (Line 22)
- Event₅: $\text{HW}(\mathbf{h}) \leq \omega$ (Line 22)

A detailed probabilistic bound for each individual condition is provided in Appendix D. Let Event_{RS} denote the overall rejection event. Therefore, the expected number of iterations of the rejection-sampling loop is approximated by Equation 1. Here, the Table 1 compares the expected number of signing iterations across parameter sets for Dilithium, MLWRSign, and ATLAS. Across all security levels, ATLAS requires noticeably fewer repetitions, averaging only 1.65 to 3.67 iterations.

$$\frac{1}{\Pr[\text{Event}_{\text{RS}}]} \approx \exp\left(n \left(\frac{\beta_1 l}{\gamma_1} + \frac{\beta_2 k}{\bar{\gamma}_2} \right)\right). \quad (1)$$

Table 1: Expected number of rejection sampling iterations per signature.

Dilithium		MLWRSign		ATLAS	
Param.	Exp.	Param.	Exp.	Param.	Exp.
Dilithium2	6.2	I (weak)	4.89	ATLAS-128	3.32
Dilithium3	5.1	II (medium)	8.89	ATLAS-256	2.72
Dilithium5	3.8	III (recomm.)	4.12	ATLAS-512	1.65
		IV (high)	5.59		
		V (very high)	3.35		
		VI (paranoia)	3.92		

Table 2: Minimum block-size b , associated Core-SVP hardness ($\log_2$), and overall gate costs ($\log_2$) for ATLAS.

		ATLAS-128	ATLAS-256	ATLAS-512
SIS	Block-size	417	909	1999
	Core-SVP (Class.)	121	265	584
	Core-SVP (Quant.)	110	241	530
SIS (SUF)	Block-size	416	868	1853
	Core-SVP (Class.)	121	253	541
	Core-SVP (Quant.)	110	230	491
Primal	Block-size	407	948	1636
	Core-SVP (Class.)	119	277	574
	Core-SVP (Quant.)	107	251	520
Dual	Block-size	406	944	1635
	Core-SVP (Class.)	118	276	478
	Core-SVP (Quant.)	107	250	432
Gate Cost (classical)		144.04	295.58	582.18

3.4 Algebraic Security

Following the security framework of Dilithium, we evaluate the concrete hardness of ATLAS against state-of-the-art lattice reduction attacks. We analyze the underlying MLWR and MSIS problems over $\mathbb{Z}_q$ via their standard unstructured counterparts, LWR and SIS, by expanding polynomials into coefficient rotation matrices in $\mathbb{Z}_q^{nk \times nl}$. For secret-key recovery, the LWR samples are converted to equivalent LWE instances with rounding error variance $\text{Var} = q^2/(12p^2)$ and solved using the primal and dual BKZ reduction attacks. For unforgeability, breaking signature correctness or producing a valid pair $(\mathbf{z}, \mathbf{u}')$ directly reduces to finding short solutions in an augmented MSIS relation $[\mathbf{A} \mid \mathbf{t}']$ bounded by admissible norms $\|\mathbf{z}\|_\infty < \gamma_1 - \beta_1$ and $\|\mathbf{u}'\|_\infty < 4_q^{\frac{p}{q}}\gamma_2$.

Concrete hardness is governed by the BKZ block-size b required to solve the Shortest Vector Problem (SVP) in dimension b , with asymptotic costs bounded by classical sieving ($2^{0.292b}$ [9]), quantum sieving ($2^{0.265b}$ [29]), and the paranoid lower bound ($2^{0.2075b}$ [21]). Table 2 details the resulting minimum block sizes and core-SVP hardness across all security levels. Factoring in dimensions-for-free ($d4f$) and practical sieving gate-count estimators following the Kyber methodology [6], Table 2 also reports the total primal gate complexities, confirming that all parameter sets exceed their target NIST security levels. We refer the reader to Appendix E for the detailed algebraic reductions, estimator settings, and the complete gate-count derivations.

4 Parameters Exploration and Cost Evaluation

4.1 Parameters Selection

Parameter selection for ATLAS involved exploring combinations of $(n, q, p, \beta_1, \beta_2, \gamma_1, \bar{\gamma}_2)$ subject to the requisite correctness and security constraints. A Python search script evaluated candidate tuples to determine optimal settings for each targeted classical security level (128, 256, and 512 bits). The resulting concrete values are summarized in Table 3.

Table 3: ATLAS Parameter Sets for Different Security Levels

Parameter Set	ATLAS-128	ATLAS-256	ATLAS-512
Classical Security	128 bits	256 bits	512 bits
Quantum Security	80 bits	128 bits	256 bits
n (Ring Degree)	128	256	512
(q, p) (Modulus)	$(2^{23}, 2^{18})$	$(2^{23}, 2^{20})$	$(2^{25}, 2^{22})$
k, l (Dimensions)	(9, 6)	(8, 7)	(8, 7)
η (noise parameter)	16	4	4
κ (challenge size)	31	60	118
(β_1, β_2) (Bounds)	(51, 8)	(73, 12)	(73, 12)
$(\gamma_1, \gamma_2, \bar{\gamma}_1, \bar{\gamma}_2)$	$(2^{19}, 2^{18}, 2^{14}, 2^{13})$	$(2^{19}, 2^{18}, 2^{16}, 2^{15})$	$(2^{21}, 2^{20}, 2^{18}, 2^{17})$
pk_{drop}	10	10	10
ω (Weight)	128	128	128

Moduli and Ring Degrees. All moduli q and p are powers of two, allowing rounding operations to be realized via bit-shifts as in Saber [19]. The ring degree $n \in \{128, 256, 512\}$ and matrix dimensions (k, l) are chosen to minimize memory and computational footprint while achieving the target security.

Noise and Challenge Set. The rounding noise parameter is fixed as $\eta := \lceil \frac{q}{2p} \rceil$. The challenge set $\text{ChSet} \subset \mathcal{R}$ contains ternary polynomials with weight κ , giving $|\text{ChSet}| = 2^\kappa \binom{n}{\kappa}$. For each parameter set, κ is set so that $|\text{ChSet}| > 2^\lambda$.

Rejection Sampling Bounds. The masking bounds satisfy $\gamma_1 = q/16$, $\bar{\gamma}_1 = \frac{p}{q}\gamma_1$, $\gamma_2 = \gamma_1/2$, and $\bar{\gamma}_2 = \bar{\gamma}_1/2$. The secret bounds β_1 and β_2 are jointly optimized over the integers to satisfy $\|\text{cs}_1\|_\infty < \beta_1$ and $\|\text{cs}_2\|_\infty < \beta_2 - 1$, while keeping the repetition exponent $n(\frac{\beta_1 l}{\gamma_1} + \frac{\beta_2 k}{\bar{\gamma}_2}) \approx 1$.

Public Key Compression and Hints. Setting $pk_{drop} = 10$ discards low-order bits from $\mathbf{t}$ to reduce key size, satisfying $\kappa(2^{pk_{drop}-1} + 1) < 2\bar{\gamma}_2 - 1$ so that truncation noise is absorbed by the verification slack. The maximum hint weight is bounded by $\omega = 128$, which holds with overwhelming probability.

4.2 Communication Cost: Key and Signature Sizes

Public Key (PK). The public key $pk = (\rho, \mathbf{t}_1)$ has size

$$|pk| = \frac{n}{8} \cdot k \cdot (\lceil \log p \rceil - pk_{drop} + 1) + 32 \quad \text{bytes},$$

where $pk_{drop} := \log(2\gamma_2) - \log(\kappa) + 1$, yielding a $\mathbf{t}_1$ bit-length of $\log p - \log(2\gamma_2) + \log(\kappa) - 1$.

Secret Key (SK). The secret key $sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{t}_0)$ comprises

$$|sk| = 112 + \frac{n}{8} \cdot (6l + pk_{drop} \cdot k) \quad \text{bytes},$$

fixing $\lceil \log(2\eta + 1) \rceil = 6$ across all parameter sets. Compared to Dilithium, ATLAS achieves an approximate 26%–34% secret key size reduction. This stems from: (i) omitting explicit storage of the noise vector $\mathbf{s}_2$, which is deterministically reconstructed under LWR during signing, and (ii) operating over modulus $p < q$, enabling a smaller pk_{drop} value than the 14 bits fixed in Dilithium.

Signature Size (Sig). The signature $\sigma = (\mathbf{z}, \mathbf{h}, c)$ requires

$$|\sigma| = \frac{n}{8} \cdot l \cdot \log(2\gamma_1) + \omega \cdot \Delta_n + k + \frac{n}{8} + \Delta_\kappa \quad \text{bytes},$$

where $\Delta_n = 1$ for $\lambda \leq 256$ and $\Delta_n = \frac{\log n}{8}$ for ATLAS-512, and $\Delta_\kappa = 8$ for $\lambda \leq 256$ and $\Delta_\kappa = \lceil \kappa/8 \rceil$ for ATLAS-512. Maintaining matching bounds on the masking vector γ_1 , challenge weight β_1 , and hint bits ω preserves signature sizes comparable to Dilithium.

Based on the parameter sets defined in Table 3, Table 4 reports the calculated communication cost for ATLAS alongside benchmarks for Dilithium [21] and MLWRSig [33].

Table 4: Communication cost (bytes) across ATLAS, Dilithium, and MLWRSign.

	ATLAS			Dilithium			MLWRSign					
	128	256	512	2	3	5	I	II	III	IV	V	VI
PK	1328	2848	6688	1312	1952	2592	896	1184	1472	1760	2336	2624
SK	2128	4016	7920	2528	4000	4864	1392	1872	2384	2864	3856	4336
Sig	2081	4656	10088	2420	3293	4595	1387	2044	2701	3358	4672	5329

5 Polynomial Multiplication

ATLAS relies heavily on polynomial arithmetic in the ring $\mathcal{R}_q$, with power-of-two modulus $q = 2^{23}, 2^{25}$ and the irreducible polynomial $x^n + 1$, where $n = 128, 256, 512$ for $\lambda = 128, 256, 512$ respectively. Addition and subtraction are simple coefficient-wise operations, but multiplication costs $\mathcal{O}(n^2)$ and dominates performance. Unlike Dilithium, ATLAS cannot use the NTT since q is not prime, so we use Karatsuba and Toom-Cook multiplication instead. Karatsuba recursively splits an n -degree multiplication into three $n/2$ -degree multiplications; once the degree reaches a threshold, a quadratic method such as schoolbook multiplication computes the product directly. Toom-Cook generalizes this by splitting an n -degree multiplication into seven $n/4$ -degree multiplications (four-way Toom-Cook, or Toom-4), with smaller multiplications handled by Karatsuba or schoolbook. Toom-4 has asymptotic complexity $\mathcal{O}(n^{\log_4 7})$, better than Karatsuba's $\mathcal{O}(n^{\log_2 3})$.

This classical approach has a drawback: each recursive base multiplication produces intermediate results of twice the degree of the sub-polynomials, requiring temporary storage for unreduced products. Repeated decomposition thus increases memory usage and coefficient rearrangement across recursion levels. To address this, we propose a striding formulation based on [10], which groups coefficients by their residue modulo the split factor: $A(x) = \sum_{i=0}^{k-1} x^i A_i(x^k)$, where $A_i(y) = \sum_{j=0}^{m-1} a_{kj+i} y^j$.

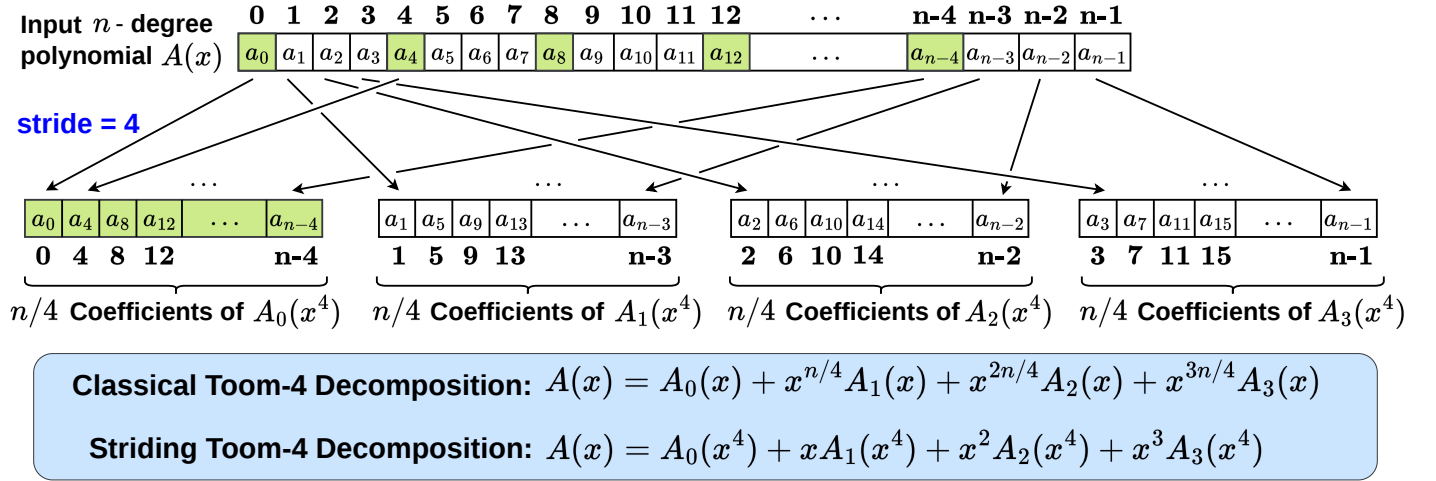


Fig. 4: Polynomial decomposition for a striding Toom-4 multiplication subroutine for n -degree input polynomial $A(x)$. Instead of a contiguous memory split, coefficients are accessed at a stride length of 4. As shown in Figure 4, a striding Toom-4 decomposition is:

$$A_0 = (a_0, a_4, a_8, \dots), A_1 = (a_1, a_5, a_9, \dots), A_2 = (a_2, a_6, a_{10}, \dots), A_3 = (a_3, a_7, a_{11}, \dots).$$

Similarly, striding Karatsuba decomposition is:

$$A(x) = A_e(x^2) + x A_o(x^2), \text{ where } A_e(y) = \sum_j a_{2j} y^j, \text{ and } A_o(y) = \sum_j a_{2j+1} y^j.$$

Toom-4 evaluation is done at points $y = \{0, \pm 1, \pm \frac{1}{2}, 2, \infty\}$, following [15]. Interpolation requires multiplication and division by scalar constants. Division by odd scalars uses their multiplicative inverses modulo 2^{32} . Since even divisors have no such

inverse, division by an even scalar $f = 2^m \cdot a$ is done in two steps: multiply by the inverse of the odd factor a , then right-shift by m positions.

In the striding formulation's interpolation phase, the reconstructed polynomial is $C(x) = \sum_{i=0}^{2k-2} X^i C_i(X^k)$, where successive interpolation polynomials $C_i(X^k)$ and $C_{i+1}(X^k)$ overlap by one power of X : the higher-degree coefficients of one overlap the lower-degree coefficients of the next. Interpolation is therefore an overlap-add operation, accumulating overlapping coefficients before the final negacyclic wraparound correction. This holds for both striding Karatsuba ($k = 2$) and striding Toom- k , differing only in the number of interpolation polynomials involved, as shown in Figure 5.

Striding Karatsuba Interpolation	Striding Toom-4 Interpolation
Input: Coefficients w_0, w_1, w_2 from classical Karatsuba interpolation Output: Result polynomial res	Input: Coefficients $w_0, \dots, w_6$ from standard Toom-4 interpolation Output: Result polynomial res
<pre> 1: Perform classical Karatsuba interpolation to obtain $[w_0, w_1, w_2]$ 2: for $i = 0$ to $n/2 - 1$ do 3: $res[2i] \leftarrow res[2i] + r_0$ 4: $res[2i + 1] \leftarrow res[2i + 1] + r_1 - r_0 - r_2$ 5: if $i = n/2 - 1$ then 6: $res[0] \leftarrow res[0] - r_2$ 7: else 8: $res[2(i + 1)] \leftarrow res[2(i + 1)] + r_2$ 9: end if 10: end for </pre>	<pre> 1: Perform classical Toom-4 interpolation to obtain $[w_0, w_1, \dots, w_6]$ 2: $prev_0 \leftarrow 0, prev_1 \leftarrow 0, prev_2 \leftarrow 0$ 3: for $i = 0$ to $n/4 - 1$ do 4: $res[4i + 3] \leftarrow r_3$ 5: if $i = 0$ then 6: $res[4i] \leftarrow r_6$ 7: $res[4i + 1] \leftarrow r_5$ 8: $res[4i + 2] \leftarrow r_4$ 9: else 10: $res[4i] \leftarrow r_6 + prev_2$ 11: $res[4i + 1] \leftarrow r_5 + prev_1$ 12: $res[4i + 2] \leftarrow r_4 + prev_0$ 13: end if 14: $prev_0 \leftarrow r_0, prev_1 \leftarrow r_1, prev_2 \leftarrow r_2$ 15: end for 16: $res[0] \leftarrow res[0] - prev_2$ 17: $res[1] \leftarrow res[1] - prev_1$ 18: $res[2] \leftarrow res[2] - prev_0$ </pre>

Fig. 5: Striding interpolation procedures for Karatsuba and Toom-4

Performance Evaluation of Different Multiplication Strategies We use a hierarchical decomposition framework to evaluate polynomial multiplication strategies for degree-128, degree-256, and degree-512 polynomials, using both classical and striding implementations. At each recursion level, a strategy (Toom-4 or Karatsuba) is selected, with schoolbook multiplication as the base case. The algorithm is parameterized by a strategy sequence specifying the method used at each recursion depth. Recursion stops at a base threshold of degree-8 or degree-16 schoolbook multiplication. At each level, the chosen strategy determines the decomposition, evaluation, and interpolation, while recursive calls compute the pointwise products.

Table 5 reports average CPU cycles for each non-striding strategy, identified by its recursive shape of tc Toom-4 layers followed by ks Karatsuba layers, down to schoolbook cutoff sb , for $n = 128, 256, 512$, with speedups relative to the pure n -degree schoolbook strategy at each n in parentheses.

Striding evaluation and interpolation change the relative performance of these strategies, as shown in Table 6. The optimal strategy for $n = 128$ remains unchanged, whereas the best configurations for $n = 256$ and $n = 512$ differ from the non-striding case. For ATLAS-256, the non-striding implementation with two levels of Toom-4 and degree-16 schoolbook multiplication performs best among the non-striding strategies. With striding, however, the combination of two levels of Toom-4, one level of Karatsuba, and degree-8 schoolbook multiplication performs better. Similarly, for the highest security level, the striding implementation with three levels of Toom-4 followed by degree-8 schoolbook multiplication outperforms both its non-striding counterpart and the other hybrid strategies. This shows that striding provides greater benefit for strategies using a degree-8 schoolbook base multiplication than for those using degree-16 schoolbook multiplication (Table 7). This behavior can be attributed to the smaller base polynomial size with $sb = 8$, which allows the operands to remain

Table 5: Average performance of non-striding multiplication strategies. Each strategy applies tc Toom-4 layers, then ks Karatsuba layers, with a schoolbook cutoff at size sb , so that $4^{tc} \cdot 2^{ks} \cdot sb = n$. Values are average CPU cycles, with speedup over pure schoolbook multiplication ($tc=0, ks=0, sb=n$) at the same n in parentheses.

$n = 128$				$n = 256$				$n = 512$			
tc	sb	ks	CPU cycles	ks	CPU cycles	ks	CPU cycles	ks	CPU cycles	ks	CPU cycles
0	n	0	21032.9 (1.00)	0	97320.6 (1.00)	0	374058.1 (1.00)				
0	8	4	14170.4 (1.48)	5	42997.1 (2.26)	6	154587.8 (2.42)				
1	8	2	10558.1 (1.99)	3	33512.3 (2.90)	4	96014.2 (3.90)				
2	8	0	7938.8 (2.65)	1	26020.1 (3.74)	2	—				
3	8		n/a		n/a	0	67448.6 (5.55)				
0	16	3	11260.0 (1.87)	4	38262.3 (2.54)	5	112607.0 (3.32)				
1	16	1	8601.4 (2.45)	2	29241.1 (3.33)	3	85933.3 (4.35)				
2	16		n/a	0	22241.1 (4.38)	1	67100.0 (5.57)				
1	32	0	—	1	—	2	115760.4 (3.23)				
2	32		n/a		n/a	0	76753.0 (4.87)				

‘—’ denotes a valid but unmeasured configuration. Avg time $\approx 0.298 \text{ ns} \times \text{cycles}$; peak memory is strategy-independent at 2496/2376/2504 KB for $n = 128/256/512$.

Table 6: Average performance of striding multiplication strategies.

n	Strategy	CPU cycles	Time (ns)	Memory (KB)
128	2Toom-4 + 8Schoolbook	7952.7	2414.4	2364
256	2Toom-4 + 16Schoolbook	26231.3	7822.4	2528
	2Toom-4 + 1Karatsuba + 8Schoolbook	23129.3	7129.8	2372
512	2Toom-4 + 1Karatsuba + 16Schoolbook	75986.1	23324.2	2552
	3Toom-4 + 8Schoolbook	57947.6	18180.9	2372

in the cache during a base-case multiplication. In contrast, $sb = 16$ requires approximately four times more memory for the multiply-accumulate operation, increasing memory re-fetches during the base multiplication, and consequently the average CPU cycle counts.

Table 7: Striding versus non-striding multiplication on matched configurations. Each row fixes the recursive shape (tc Toom-4 layers, then ks Karatsuba layers, schoolbook cutoff at size sb) and varies only the memory access pattern. Speed-up is non-striding cycles divided by striding cycles; values above 1 favour striding.

n	tc	ks	sb	Avg CPU cycles			Memory (KB)	
				Non-striding	Striding	Speed-up	Non-striding	Striding
128	2	0	8	7938.8	7952.7	1.00	2496	2364
256	2	0	16	22241.1	26231.3	0.85	2376	2528
256	2	1	8	26020.1	23129.3	1.12	2376	2372
512	2	1	16	67100.0	75986.1	0.88	2504	2552
512	3	0	8	67448.6	57947.6	1.16	2504	2372

6 Performance of ATLAS

We provide a reference and an AVX2-optimized implementation for x86-64 platforms. Performance is evaluated on an AMD Ryzen 7 PRO 8840HS processor running Ubuntu 22.04 LTS. All implementations were written in C/C++ and compiled with GCC 11.4.0 (-O2 for reference and -O3 for AVX2-optimized builds). CPU cycles were captured via hardware performance counters across 1,000 runs per operation, and average cycle counts are reported.

Reference and Optimized Implementation The *reference implementation* avoids secret-dependent branching and memory accesses wherever long-term secret material is involved. Since ATLAS’s moduli are powers of two, modular reduction uses only bitmasking ($x \& (q-1)$), removing the need for division, Montgomery, or Barrett reduction. All supporting

Table 8: Performance of ATLAS (Reference vs. Optimized) for KeyGen, Sign, and Verify under SM3, SHAKE, and TurboSHAKE. All figures are in thousands of clock cycles (median/average)

Level	Impl.	SM3			SHAKE			TurboSHAKE		
		KeyGen	Sign	Verify	KeyGen	Sign	Verify	KeyGen	Sign	Verify
ATLAS-128	Ref	1462.2/1472.0	3734.6/4278.0	1542.3/1553.3	473.7/479.3	1378.7/1647.0	506.5/515.3	366.1/371.1	1286.7/1568.6	404.8/408.1
	Opt	1848.4/1868.9	3912.8/4438.4	1909.2/1943.5	216.1/220.2	734.5/868.8	225.4/230.1	207.1/209.2	699.4/825.4	219.8/222.3
ATLAS-256	Ref	2511.2/2525.9	6495.3/7611.6	2686.9/2713.4	920.5/919.4	2519.7/2953.6	981.4/981.3	809.9/821.0	2451.6/2895.2	874.1/892.7
	Opt	2473.2/2505.8	5115.5/5960.9	2565.8/2589.8	400.4/404.8	1135.9/1347.9	411.6/419.5	386.5/391.4	1103.3/1319.0	399.7/407.1
ATLAS-512	Ref	7191.7/7215.7	14434.1/18710.0	7719.9/7763.5	2920.3/2982.1	6004.4/7457.8	3162.5/3196.3	2588.8/2628.7	5623.2/7052.9	2831.5/2862.5
	Opt	6413.7/6479.6	11471.6/14524.4	6614.8/6702.0	2220.4/2220.9	4805.9/6018.3	2420.4/2399.6	2127.2/2147.7	4649.1/5837.5	2329.9/2355.3

arithmetic is branch-free. Constant-time execution is not enforced where the revealed information is independent of the secret key. For example, signing’s rejection conditions may reveal which condition triggered, since the rejection probability depends only on the public parameters γ_1 and β_1 . Likewise, revealing the challenge $c \leftarrow H(\mu \parallel \mathbf{w}_1)$ during rejected signing attempts is safe, since $\mathbf{w}_1$ comes from the public masking vector $\mathbf{y}$ and has high min-entropy.

In *optimized implementation* the degree-8 schoolbook multiplication and the core signing/verification routines (mainly supporting algorithms and polynomial arithmetic) use AVX2 intrinsics to process multiple 32-bit coefficients per instruction. Next, polynomial multiplication uses a strided hierarchical decomposition based on Toom-Cook and Karatsuba multiplication. The best configurations found were: two levels of Toom-4 followed by degree-8 schoolbook multiplication for $n = 128$; two levels of Toom-4, one level of Karatsuba, and degree-8 schoolbook multiplication for $n = 256$; and three levels of Toom-4 followed by degree-8 schoolbook multiplication for $n = 512$.

The reference implementation expands $\mathbf{A}$ from seed ρ one (i, j) index at a time via SHAKE-128. The optimized implementation instead batches four indices per call via `expand_mat_4x`, using SHAKE-128x4 [12] to run four Keccak- $f[1600]$ permutations in parallel over AVX2 SIMD lanes (with a scalar routine for the residual tail). This cuts permutation calls roughly $4\times$ without changing the sampled coefficient distribution.

Performance Analysis, Considerations and Trade-offs Table 8 shows that hash-function choice has a large impact on ATLAS’s performance, and that the AVX2 optimizations do not benefit all hash functions equally. With SHAKE and TurboSHAKE, the optimized implementation is substantially faster than the reference at 128 and 256 bits: `sig_keygen`, `sig_sign`, and `sig_verify` all improve by roughly $1.8\text{--}2.4\times$ (around 45%–58% cycle reduction) at these two levels. SM3 behaves differently: at 128 bits, the optimized implementation is actually slower than the reference across all three operations, since SM3 hashing (unlike SHAKE and TurboSHAKE [13]) is not AVX2-accelerated in our implementation and its cost comes to dominate the small polynomial-arithmetic savings; only at 256 and 512 bits, where polynomial arithmetic accounts for a larger share of the runtime, does the optimized implementation regain a modest edge (roughly 5%–27%). Across all three hash functions, TurboSHAKE consistently yields the lowest absolute cycle counts, followed by SHAKE, with SM3 the slowest by a wide margin, reflecting the relative cost of each hash function’s internal compression function rather than any property of ATLAS itself.

Table 9 places ATLAS-128 (SHAKE, optimized) in the context of other post-quantum signature schemes at comparable security. Among lattice-based schemes, ATLAS’s optimized `sig_keygen` (216K cycles) and `sig_verify` (225K cycles) are competitive with Dilithium2 and MLWRSign-III and far cheaper than Falcon-512, whose floating-point Gaussian sampling drives keygen above 23M cycles. ATLAS’s `sig_sign` (734K cycles) sits between Dilithium3 and MLWRSign-III, reflecting the cost of rejection sampling common to all Fiat-Shamir-with-aborts schemes. Against non-lattice candidates, ATLAS is faster across all three operations than the hash-based SPHINCS⁺ variants and the MPCitH-based MAYO, MQOM, FAEST, and SDitH schemes, and faster than the isogeny-based SQISign, whose sign and verify costs run into the millions of cycles. These results indicate that ATLAS offers a favourable balance of signing, verification, and key-generation cost relative to both lattice and non-lattice post-quantum signature families.

Discussions on Side-channel Security A primary source of timing side-channel leakage in lattice-based schemes is variable-time operations such as rejection sampling over a prime modulus. Using power-of-two moduli allows us to implement all modular reductions with simple bitmasking operations without data-dependent branching, helping us avoid variable-time operations. The implementations of ATLAS also do not use any conditional jump operation based on secret data, use secret data-dependent branches, or lookup tables to avoid timing attacks [1]. Although our implementation does not protect against power side-channel attacks, it can be secured through masking techniques. The use of power-of-two

Table 9: Comparison of CPU cycle counts (in thousands of cycles, K) across post-quantum signature schemes (NIST Level 1 / Level 2 benchmarks).

Scheme	Type (Security)	KeyGen (10^3)		Sign (10^3)		Verify (10^3)	
		Ref	Opt/AVX2	Ref	Opt/AVX2	Ref	Opt/AVX2
<i>Lattice-based Signatures</i>							
MLWRSign-III [33]	Lattice (123)	456	218	1,412	757	499	256
Dilithium2 [20]	Lattice (L2)	300	124	1,081	259	327	118
Falcon-512 [22]	Lattice (L1)	23,381	27,128	898	974	122	122
HAETAE-2	Lattice/Bimodal (L1)	480	210	1,720	680	440	190
<i>Hash-based Signatures (FIPS 205) [32]</i>							
SPHINCS ⁺ -128s-shake	Hash / Stateless (L1)	616,484	143,901	4,682,571	1,102,471	4,764	1,189
SPHINCS ⁺ -128f-shake	Hash / Stateless (L1)	9,649	2,249	239,794	56,934	12,910	3,346
<i>Non-Lattice NIST On-Ramp Candidates [5]</i>							
MAYO-1 [14]	Multivariate (L1)	4,951	195	2,851	185	966	112
MQOM-L1 [11]	MPCitH/MQ (L1)	110	90	1,830	1,150	1,730	1,120
FAEST-128s[8]	MPCitH/AES (L1)	11,148	77	239,794	1,700	20,335	1,200
SDitH-L1-fast[3]	MPCitH/Code (L1)	210	66	24,500	2,080	23,800	1,780
SQISign-Level1 [2]	Isogeny (L1)	43,900	32,200	125,200	93,900	18,100	12,100
<i>Proposed Scheme (ATLAS)</i>							
ATLAS-128 (this paper)	Lattice (L1)	473	216	1,378	734	506	225

moduli and the MLWR problem in ATLAS eliminates the necessity for costly modular reductions and rejection sampling in the primary arithmetic path. This choice facilitates efficient arbitrary-order masking by making computationally expensive operations, such as Arithmetic-to-Boolean conversions, significantly cheaper compared to schemes that utilize prime moduli (e.g., Dilithium). Previous works have proven this result for the Key Encapsulation Mechanism (KEM), and it can potentially be extended to digital signatures [16,25,27]. The exact cost of implementing masking in ATLAS is an interesting topic that will be explored in future work.

7 Conclusions

We introduced ATLAS, an MLWR-based signature scheme built on the Fiat–Shamir with aborts framework. By replacing explicit noise sampling with deterministic rounding, ATLAS eliminates the error term used in schemes like Dilithium and HAETAE, yielding smaller secret keys and lower key generation cost, while remaining structurally close enough to Dilithium to inherit its cryptanalytic foundations. We provided parameter sets for three security levels (128, 256, and 512 bits), including, to our knowledge, the first MLWR-based parameter set at 512-bit security. Power-of-two moduli enable efficient bitwise modular reduction and avoid rejection sampling, and we designed a hierarchical, AVX2-vectorized Toom-Cook/Karatsuba multiplication strategy to support this choice. We also showed ATLAS can be instantiated with either SHAKE or KDF-SM3, with quantified overhead between the two. Overall, ATLAS offers a compact and efficient alternative among lattice-based signatures. Future work includes further security analysis, multiplication optimizations, and additional hardware targets.

References

1. TIMECOP: Automatic detection of timing side channels. <https://www.post-apocalyptic-crypto.org/timecop/> (2020), accessed: 2026-06-27
2. Aardal, M.A., Adj, G., Aranha, D.F., Basso, A., Borin, G., Martínez, I.A.C., Chávez-Saab, J., Santos, M.C.R., Dartois, P., De Feo, L., Duparc, M., Espitau, T., Eriksen, J.K., Fouotsa, T.B., Gazzoni Filho, D.L., Hess, B., Invernizzi, R., Kohel, D., Leroux, A., Longa, P., Maino, L., Meyer, M., Mula, M., Nakagawa, K., Onuki, H., Panny, L., Patranabis, S., Petit, C., Pope, G., Reijnders, K., Robert, D., Rodríguez-Henríquez, F., Schaeffler, S., Vercauteren, F., Wallet, A., Wesolowski, B., van Woerden, W.: SQISIGN: Algorithm specifications and supporting documentation. Technical report, NIST post-quantum cryptography standardization process (additional signatures), National Institute of Standards and Technology (2026), <https://csrc.nist.gov/projects/pqc-dig-sig>
3. Aguilar Melchor, C., Bettaleb, S., Bidoux, L., Feneuil, T., Gaborit, P., Gama, N., Gueron, S., Howe, J., Hülsing, A., Joseph, D., Joux, A., Kulkarni, M., Persichetti, E., Randrianarisoa, T.H., Rivain, M., Yue, D.: The syndrome decoding in the head (SD-in-the-Head) signature scheme. Technical report, NIST post-quantum cryptography standardization process (additional signatures), National Institute of Standards and Technology (2026), <https://csrc.nist.gov/projects/pqc-dig-sig>
4. Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Liu, Y.K., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D.: Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process. Online. Accessed 26th January, 2024 (2022), <https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413-upd1.pdf>
5. Alagic, G., Bros, M., Ciadoux, P., Dang, Q., Dang, T.H., Kelsey, J., Lichtinger, J., Liu, Y.K., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Silberg, H., Smith-Tone, D., Waller, N.: Status Report on the Second Round of the Additional Digital Signature Schemes for the NIST Post-Quantum Cryptography Standardization Process. NIST Internal Report (IR) NIST IR 8610, National Institute of Standards and Technology, Gaithersburg, MD (2026). <https://doi.org/10.6028/NIST.IR.8610>
6. Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Kyber (version 3.02) – Submission to round 3 of the NIST post-quantum cryptography standardization process. Available online at <https://pq-crystals.org> (August 2021)
7. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) *Advances in Cryptology – EUROCRYPT 2012*. pp. 719–737. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
8. Baum, C., Beullens, W., Braun, L., Delpech de Saint Guilhem, C., Kloof, M., Majenz, C., Mukherjee, S., Orsini, E., Ramacher, S., Rechberger, C., Roy, L., Scholl, P.: FAEST: Algorithm specifications. Technical report, NIST post-quantum cryptography standardization process (additional signatures), National Institute of Standards and Technology (2026), <https://csrc.nist.gov/projects/pqc-dig-sig>
9. Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New directions in nearest neighbor searching with applications to lattice sieving. In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*. p. 10–24. SODA '16, Society for Industrial and Applied Mathematics, USA (2016)
10. Becker, H., Mera, J.M.B., Karmakar, A., Yiu, J., Verbaauwhede, I.: Polynomial multiplication on embedded vector architectures. *Cryptology ePrint Archive*, Paper 2021/998 (2021), <https://eprint.iacr.org/2021/998>
11. Benadjila, R., Feneuil, T., Bouillaguet, C., Rivain, M.: MQOM: MQ on my mind. Technical report, NIST post-quantum cryptography standardization process (additional signatures), National Institute of Standards and Technology (2026), <https://csrc.nist.gov/projects/pqc-dig-sig>
12. Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., Van Keer, R.: The eXtended Keccak Code Package (XKCP). <https://github.com/XKCP/XKCP>, see `lib/low/KeccakP-1600-times4/AVX2/`, the four-way vectorized Keccak- $p[1600]$ permutation used for parallel SHAKE-128/SHAKE-256
13. Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., Van Keer, R., Viguier, B.: Kangarootwelve and turboshake. RFC 9861 (Oct 2025). <https://doi.org/10.17487/RFC9861>, <https://www.rfc-editor.org/rfc/rfc9861>
14. Beullens, W., Campos, F., Celi, S., Hess, B., Kannwischer, M.J.: MAYO. Technical report, NIST post-quantum cryptography standardization process (additional signatures), National Institute of Standards and Technology (2026), <https://csrc.nist.gov/projects/pqc-dig-sig>
15. Bodrato, M., Zanzi, A.: Integer and polynomial multiplication: towards optimal toom-cook matrices. In: *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation*. p. 17–24. ISSAC '07, Association for Computing Machinery, New York, NY, USA (2007). <https://doi.org/10.1145/1277548.1277552>, <https://doi.org/10.1145/1277548.1277552>
16. Bronchain, O., Cassiers, G.: Bitslicing arithmetic/boolean masking conversions for fun and profit with application to lattice-based kems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2022**(4), 553–588 (2022). <https://doi.org/10.46586/TCHES.V2022.I4.553-588>, <https://doi.org/10.46586/tches.v2022.i4.553-588>
17. Cheon, J.H., Choe, H., Devevey, J., Güneysu, T., Hong, D., Krausz, M., Land, G., Shin, J., Stehlé, D., Yi, M.: Haetae: Hyperball bimodal module rejection signature scheme. Supporting Document of Korean Post-Quantum Cryptography Competition (2023)
18. Cremers, C., Dölzl, S., Fiedler, R., Fischlin, M., Janson, C.: BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. In: *42nd IEEE Symposium on Security and Privacy (S&P)*. pp. 1696–1714. IEEE (2021)
19. D’Anvers, J.P., Karmakar, A., Sinha Roy, S., Vercauteren, F.: Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) *Progress in Cryptology – AFRICACRYPT 2018*. pp. 282–305. Springer International Publishing, Cham (2018)

20. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium: Algorithm specifications and supporting documentation. Tech. rep., NIST Post-Quantum Cryptography Standardization, Round 3 Submission (Oct 2020), <https://pq-crystals.org/dilithium/data/dilithium-specification-round3.pdf>, accessed: 2026-09-06
21. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems* **2018**(1), 238–268 (Feb 2018). <https://doi.org/10.13154/tches.v2018.i1.238-268>, <https://tches.iacr.org/index.php/TCHES/article/view/839>
22. Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: Falcon: Fast-Fourier lattice-based compact signatures over NTRU – specifications v1.2. Tech. rep., NIST (2020), <https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions>
23. Kiltz, E., Lyubashevsky, V., Schaffner, C.: A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In: *Advances in Cryptology – EUROCRYPT 2018. Lecture Notes in Computer Science*, vol. 10822, pp. 552–586. Springer (2018)
24. KpqC Research Center: Selected algorithms from the KpqC competition round 2. Post-Quantum Cryptography Research Center, South Korea (January 2025), https://www.kpqc.or.kr/competition_02.html, official Announcement of National PQC Standards
25. Kundu, S., D’Anvers, J., Beirendonck, M.V., Karmakar, A., Verbaauwhede, I.: Higher-order masked saber. In: Galdi, C., Jarecki, S. (eds.) *Security and Cryptography for Networks - 13th International Conference, SCN 2022, Amalfi, Italy, September 12-14, 2022, Proceedings. Lecture Notes in Computer Science*, vol. 13409, pp. 93–116. Springer (2022). https://doi.org/10.1007/978-3-031-14791-3_5, https://doi.org/10.1007/978-3-031-14791-3_5
26. Kundu, S., Ghosh, A., Karmakar, A., Sen, S., Verbaauwhede, I.: Rudraksh: A compact and lightweight post-quantum key-encapsulation mechanism. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2025**(2), 647–680 (2025). <https://doi.org/10.46586/TCHES.V2025.I2.647-680>, <https://doi.org/10.46586/tches.v2025.i2.647-680>
27. Kundu, S., Karmakar, A., Verbaauwhede, I.: On the masking-friendly designs for post-quantum cryptography. In: Regazzoni, F., Mazumdar, B., Parameswaran, S. (eds.) *Security, Privacy, and Applied Cryptography Engineering - 13th International Conference, SPACE 2023, Roorkee, India, December 14-17, 2023, Proceedings. Lecture Notes in Computer Science*, vol. 14412, pp. 162–184. Springer (2023). https://doi.org/10.1007/978-3-031-51583-5_10, https://doi.org/10.1007/978-3-031-51583-5_10
28. Kundu, S., Norga, Q., Karmakar, A., Gangopadhyay, S., Mera, J.M.B., Verbaauwhede, I.: Scabbard: An exploratory study on hardware aware design choices of learning with rounding-based key encapsulation mechanisms. *ACM Trans. Embed. Comput. Syst.* **24**(1), 10:1–10:40 (2025). <https://doi.org/10.1145/3696208>, <https://doi.org/10.1145/3696208>
29. Laarhoven, T.: Search problems in cryptography: from fingerprinting to lattice sieving. Phd thesis 1 (research tu/e / graduation tu/e), Mathematics and Computer Science (Feb 2016), proefschrift
30. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography* **75**(3), 565–599 (2015). <https://doi.org/10.1007/s10623-014-9938-4>
31. National Institute of Standards and Technology: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Federal Information Processing Standards Publication (FIPS PUB) 202 (2015). <https://doi.org/10.6028/NIST.FIPS.202>
32. NIST: FIPS 205 Stateless Hash-Based Digital Signature Standard. Online. Accessed 10th November, 2024 (2024), <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.pdf>
33. Okada, H., Takayasu, A., Fukushima, K., Kiyomoto, S., Takagi, T.: A compact digital signature scheme based on the module-lwr problem. In: Meng, W., Gollmann, D., Jensen, C.D., Zhou, J. (eds.) *Information and Communications Security*. pp. 73–90. Springer International Publishing, Cham (2020)
34. Ravi, P., Roy, D.B., Bhasin, S., Chattopadhyay, A., Mukhopadhyay, D.: Number "not used" once - practical fault attack on pqm4 implementations of NIST candidates. In: Polian, I., Stöttinger, M. (eds.) *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings. Lecture Notes in Computer Science*, vol. 11421, pp. 232–250. Springer (2019). https://doi.org/10.1007/978-3-030-16350-1_13, https://doi.org/10.1007/978-3-030-16350-1_13
35. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* **56**(6) (Sep 2009). <https://doi.org/10.1145/1568318.1568324>, <https://doi.org/10.1145/1568318.1568324>
36. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)* **56**(6), 1–40 (2009). <https://doi.org/10.1145/1568318.1568324>
37. Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming* **66**(1), 181–199 (Aug 1994). <https://doi.org/10.1007/BF01581144>, <https://doi.org/10.1007/BF01581144>
38. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: *Proceedings 35th annual Symposium on Foundations of Computer Science*. pp. 124–134. Ieee (1994)
39. State Commercial Cryptography Administration: Specification of sm3 cryptographic hash algorithm. Tech. rep., Office of Security Commercial Code Administration (2010), <https://oscca.gov.cn/sca/xxgk/2010-12/17/1002389/files/302a3ada057c4a73830536d03e683110.pdf>, official Standard Release

8 Appendix

A Descriptions Supporting Algorithms

decompose_keygen. This Algorithm (see Figure 1) partitions a coefficient $a \in \mathbb{Z}_q$ into a high-order component a_1 and a low-order remainder a_0 such that $a \equiv a_1 \cdot 2^{pk_{drop}} + a_0 \pmod{q}$. The high part a_1 is computed via a biased rounding $a_1 = \left\lfloor \frac{a-1+2^{pk_{drop}-1}}{2^{pk_{drop}}} \right\rfloor$, which centres the decomposition to ensure a_0 remains small. The remainder is then derived as $a_0 = q + a - a_1 \cdot 2^{pk_{drop}}$.

decompose_sign. Decomposes (see Figure 1) a coefficient $a \in \mathbb{Z}_p$ into a high-order component a_1 and a low-order remainder a_0 such that $a \equiv a_1 \cdot 2^\delta + a_0 \pmod{p}$, where $\delta = \text{bits}_{\bar{\gamma}_1}$. The high part is computed using a biased rounding shift $a_1 = \left\lfloor \frac{a-1+\bar{\gamma}_2}{2^\delta} \right\rfloor$. This bias ensures that the resulting low part a_0 is centred, minimizing its magnitude. The remainder is defined as $a_0 = p + a - a_1 \cdot 2^\delta$ to ensure a positive representation in modulo p . Finally, the high part is constrained to the quotient space via $a_1 \leftarrow a_1 \bmod (\frac{p}{2^\delta})$, implemented efficiently as a bitwise AND: $a_1 \& ((p \gg \delta) - 1)$. Bounding a_0 through this decomposition is necessary for both protocol correctness and the security of the abort condition.

make_hint. Determines (see Figure 1) whether adding a value $b \in \mathbb{Z}_p$ to a coefficient $a \in \mathbb{Z}_p$ changes its high-bit decomposition. Let $\tilde{a} = (a + b) \bmod p$, which is implemented in the code as $(a + b) \& (p - 1)$. The function computes the high parts using the decomposition procedure, $(a_1, _) \leftarrow \text{decompose_sign}(a)$, and $(a_2, _) \leftarrow \text{decompose_sign}(\tilde{a})$. The output is defined as

$$\text{make_hint}(a, b) = \begin{cases} 1, & \text{if } a_1 \neq a_2, \\ 0, & \text{otherwise.} \end{cases}$$

Thus, the function outputs a single-bit hint indicating whether the addition of b causes a change in the high part of a . This hint is used in the signing procedure to assist in reconstructing the correct high bits after perturbation, while ensuring that no additional information about the secret is leaked.

use_hint. The Algorithm (see Figure 1) reconstructs the correct high part of a coefficient using a one-bit hint. It takes $a \in \mathbb{Z}_p$ and $\text{hint} \in \{0, 1\}$ as input. First, the coefficient is reduced in modulo p and decomposed as $(a_1, a_0) \leftarrow \text{decompose_sign}(a \bmod p)$, such that $a \equiv a_1 \cdot 2^\delta + a_0 \pmod{p}$, where $\delta = \text{bits}_{\bar{\gamma}_1}$. The notation $\text{bits}_{\bar{\gamma}_1}$ denote the number of bits required to represent the value $\bar{\gamma}_1$, that is, $\lceil \log_2 \bar{\gamma}_1 \rceil$. If $\text{hint} = 0$, the function returns a_1 , indicating that no correction is needed. If $\text{hint} = 1$, the function adjusts the high part based on the sign of the low part a_0 . Since a_0 is represented in modulo p as $a_0 = p + (a - a_1 \cdot 2^\delta)$, values satisfying $a_0 < p$ correspond to negative representatives, while values $a_0 \geq p$ correspond to non-negative representatives. Therefore, the update rule is,

$$a_1 \leftarrow \begin{cases} (a_1 + 1) \bmod 2^{\delta_1}, & \text{if } a_0 > p, \\ (a_1 - 1) \bmod 2^{\delta_1}, & \text{otherwise,} \end{cases}$$

where $2^{\delta_1} = \frac{p}{2^\delta}$ and the modular reduction is implemented via bit-masking $\& ((p \gg \delta) - 1)$. Thus, the function uses the hint to correct the high part by ± 1 , allowing recovery of the correct decomposition after perturbation during signing, while preserving security.

B Proof of Theorem 1 (Correctness)

Theorem 1 (Correctness). *The $\text{ATLAS} = \{\text{sig_keygen}, \text{sig_sign}, \text{sig_verify}\}$ signature scheme is correct if, for any signature $\sigma = (\mathbf{z}, \mathbf{h}, c)$ generated by $\text{sig_sign}(sk, M)$, the reconstructed high-order bits $\mathbf{w}'_1$ satisfy $\mathbf{w}'_1 = \mathbf{w}_1$. This equality is guaranteed if the rejection sampling criteria in sig_sign ensure that the error terms associated with $\mathbf{s}_1, \mathbf{s}_2$, and $\mathbf{t}_0$ remain within the correction range of the use_hint function.*

Proof. To prove the correctness of the signature scheme, we evaluate the reconstruction of the high-order bits during verification. Given that $\mathbf{ct}$ contains only integer coefficients, we move the term inside the rounding operator to expand the relation:

$$\begin{aligned} \left\lfloor \frac{p}{q} \mathbf{Az} \right\rfloor - \mathbf{ct} &= \left\lfloor \frac{p}{q} \mathbf{A}(\mathbf{y} + c\mathbf{s}_1) - \mathbf{ct} \right\rfloor \\ &= \left\lfloor \frac{p}{q} \mathbf{Ay} - c \left(\mathbf{t} - \frac{p}{q} \mathbf{As}_1 \right) \right\rfloor. \end{aligned} \quad (2)$$

By applying the definition of the secret key component $\mathbf{s}_2 = \mathbf{t} - \frac{p}{q} \mathbf{As}_1$ and incorporating the rounding errors $\xi_1 = \left\lfloor \frac{p}{q} \mathbf{Ay} \right\rfloor - \frac{p}{q} \mathbf{Ay}$ and $\xi_2 = \lfloor c\mathbf{s}_2 \rfloor - c\mathbf{s}_2$, the expression simplifies to:

$$\begin{aligned} \left\lfloor \frac{p}{q} \mathbf{Az} \right\rfloor - \mathbf{ct} &= \left\lfloor \frac{p}{q} \mathbf{Ay} - c\mathbf{s}_2 \right\rfloor \\ &= \left\lfloor \left\lfloor \frac{p}{q} \mathbf{Ay} \right\rfloor - \left\lfloor \frac{p}{q} \mathbf{Ay} \right\rfloor + \frac{p}{q} \mathbf{Ay} \right. \\ &\quad \left. - \lfloor c\mathbf{s}_2 \rfloor + \lfloor c\mathbf{s}_2 \rfloor - c\mathbf{s}_2 \right\rfloor \\ &= \left\lfloor \left\lfloor \frac{p}{q} \mathbf{Ay} \right\rfloor - \xi_1 - \lfloor c\mathbf{s}_2 \rfloor + \xi_2 \right\rfloor \\ &= \left\lfloor \frac{p}{q} \mathbf{Ay} \right\rfloor - \lfloor c\mathbf{s}_2 \rfloor + \lfloor \xi_2 - \xi_1 \rfloor \\ \implies \left\lfloor \frac{p}{q} \mathbf{Az} \right\rfloor - \mathbf{ct} &= \mathbf{w} - \lfloor c\mathbf{s}_2 \rfloor + \boldsymbol{\nu}, \end{aligned} \quad (3)$$

where $\mathbf{w} = \left\lfloor \frac{p}{q} \mathbf{Ay} \right\rfloor$ and $\boldsymbol{\nu} = \lfloor \xi_2 - \xi_1 \rfloor$. Substituting the public key decomposition $\mathbf{t} = \mathbf{t}_1 \cdot 2^d + \mathbf{t}_0$, we derive the input to the hint function as $\left\lfloor \frac{p}{q} \mathbf{Az} \right\rfloor - \mathbf{ct}_1 \cdot 2^d = \mathbf{w} - \lfloor c\mathbf{s}_2 \rfloor + \boldsymbol{\nu} + \mathbf{ct}_0$. Provided the noise remains within the supported boundaries of the hint mechanism, the verifier reconstructs $\mathbf{w}'_1 = \text{decompose_sign}(\mathbf{w} - \lfloor c\mathbf{s}_2 \rfloor + \boldsymbol{\nu}, \bar{\gamma}_2)$. Since the signing algorithm enforces the condition $\mathbf{r}_1 = \mathbf{w}_1$, the identity $\mathbf{w}'_1 = \mathbf{w}_1$ holds. This completes the proof.

C Proof of EUF-KOA Security

In this appendix, we provide the complete proof of Lemma 1, establishing the EUF-KOA security of the proposed ATLAS signature scheme. The proof of Theorem 1 can be established using techniques analogous to those employed in Dilithium [21]. For convenience, we restate the lemma.

We first formally define the EUF-KOA security notions.

Definition 4 (EUF-KOA Security). A digital signature scheme DS is Existentially Unforgeable under Key-Only Attack (EUF-KOA) if it is computationally infeasible for an adversary $\mathcal{A}$ to produce a valid forgery given only the public key. Unlike the SUF-CMA model, $\mathcal{A}$ lacks access to a signing oracle $\text{OSign}(sk, \cdot)$, effectively setting the query set $\mathcal{Q} = \emptyset$. The advantage of an adversary $\mathcal{A}$ is defined as:

$$\text{Adv}_{\text{DS}}^{\text{EUF-KOA}}(\mathcal{A}) := \Pr \left[\text{Verify}(pk, \mu^*, \sigma^*) = 1 \mid \begin{array}{l} (pk, sk) \leftarrow \text{KeyGen}(1^\lambda) \\ (\mu^*, \sigma^*) \leftarrow \mathcal{A}(pk) \end{array} \right]$$

Lemma 1 (EUF-KOA SECURITY). Suppose $\mathcal{QA}$ is a quantum adversary against the EUF-KOA-security of the ATLAS signature scheme that issues q_H queries to the quantum random oracle $|H\rangle$. Then, there exist quantum adversaries $\mathcal{B}$ and $\mathcal{C}$ that solve the MLWR problem with parameters $\text{pp}_1 = \{k, l, S\}$ and the SelfTargetMSIS problem with parameters $\text{pp}_2 = \{H, k, l+1, \zeta\}$, respectively, such that the following holds.

$$\text{Adv}_{\text{ATLAS}}^{\text{EUF-KOA}}(\mathcal{QA}) \leq \text{Adv}_{\text{MLWR}}^{\text{pp}_1}(\mathcal{B}) + \text{Adv}_{\text{SelfTargetMSIS}}^{\text{pp}_2}(\mathcal{C})$$

Furthermore, the time complexity is bounded by $\text{Time}(\mathcal{B}) = \text{Time}(\mathcal{C}) = \text{Time}(\mathcal{QA}) + q_H$.

Proof. We prove the lemma through a sequence of hybrid games, Game_0 and Game_1 , each differing from the previous in exactly one step. We denote $\Pr[\text{Game}_i]$ for the probability that $\mathcal{QA}$ wins Game_i .

Game₀ (The original EUF-KOA game). The challenger generates a **ATLAS** public key $pk = (\mathbf{A}, \mathbf{t})$ by sampling $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{k \times l}$, $\mathbf{s}_1 \xleftarrow{\$} \mathcal{S}$, and computing $\mathbf{t} = \lfloor \frac{p}{q} \mathbf{A} \mathbf{s}_1 \rfloor \in \mathcal{R}_p^k$. Then pk is given to $\mathcal{QA}$, who may query the quantum random oracle $|\mathbf{H}\rangle$ up to q_H times. $\mathcal{QA}$ wins if it outputs a valid forgery (M, σ) with $\sigma = (\mathbf{z}, \mathbf{h}, c)$ that passes the verification check and whose message M was never signed. By definition,

$$\Pr[\text{Game}_0] = \text{Adv}_{\text{ATLAS}}^{\text{EUF-KOA}}(\mathcal{QA}).$$

Game₁ (Switch to a pseudorandom public key). In this game, we replace the honestly generated $\mathbf{t}$ with the high-order part of a uniformly random $\mathbf{t}' \xleftarrow{\$} \mathcal{R}_q^k$. Concretely, the challenger:

1. Receives an **SelfTargetMSIS** instance $[\mathbf{A}, \mathbf{t}'] \in \mathcal{R}_q^{k \times l} \times \mathcal{R}_q^k$ from the **SelfTargetMSIS** challenger.
2. Decomposes $\mathbf{t}'$ into its high- and low-order parts: $\mathbf{t}' = \frac{q}{p} \mathbf{t} + \mathbf{v}$, where $\mathbf{t} \in \mathcal{R}_p^k$ and $\mathbf{v} \in \mathcal{R}_{q/p}^k$. Because $\mathbf{t}'$ is uniform over $\mathcal{R}_q^k$, the distribution of $\mathbf{t}$ is also uniform over $\mathcal{R}_p^k$.
3. Sends $(\mathbf{A}, \mathbf{t})$ to $\mathcal{QA}$ as the public key and simulates the quantum random oracle $|\mathbf{H}\rangle$.

Transition analysis. The only difference between Game_0 and Game_1 is that the public key is now derived from a uniform $\mathbf{t}' \in \mathcal{R}_q^k$ rather than being computed via the **MLWR** rounding step. By the hardness of $\text{MLWR}_{\text{pp}_1}$, these two distributions are computationally indistinguishable. Any adversary $\mathcal{QA}$ that distinguishes them can be used to build an **MLWR** solver $\mathcal{B}$, so

$$|\Pr[\text{Game}_1] - \Pr[\text{Game}_0]| \leq \text{Adv}_{\text{MLWR}}^{\text{pp}_1}(\mathcal{B}).$$

Game₂ (Reducing a forgery to an SelfTargetMSIS solution). This game is identical to Game_1 . However, we now show that any valid forgery $\mathcal{QA}$ produces can be converted into a solution of the underlying **SelfTargetMSIS**_{pp₂} instance, so the winning probability transfers to $\mathcal{C}$.

Suppose $\mathcal{QA}$ outputs a valid forgery (M, σ) with $\sigma = (\mathbf{z}, \mathbf{h}, c)$. By the correctness of the verification algorithm, the forgery satisfies $c = \mathbf{H}(\mu \parallel \mathbf{w}_1)$, where $\mathbf{w}_1 = \text{use_hint}\left(\mathbf{h}, \left\lfloor \frac{p}{q} \mathbf{A} \mathbf{z} \right\rfloor - c \mathbf{t}_1 \cdot 2^{pk_{\text{drop}}}, 2\bar{\gamma}_2\right)$, and the norm constraints $\|\mathbf{z}\|_\infty < \gamma_1 - \beta_1$ and $\text{HW}(\mathbf{h}) < \omega$ are satisfied.

Step 1: Unpacking use_hint. By definition of use_hint , for a vector $\mathbf{u}$ with $\|\mathbf{u}\|_\infty \leq 2\bar{\gamma}_2 + 1$ we can write

$$2\bar{\gamma}_2 \cdot \text{use_hint}\left(\mathbf{h}, \left\lfloor \frac{p}{q} \mathbf{A} \mathbf{z} \right\rfloor - c \mathbf{t}_1 \cdot 2^{pk_{\text{drop}}}, 2\bar{\gamma}_2\right) = \left\lfloor \frac{p}{q} \mathbf{A} \mathbf{z} \right\rfloor - c \mathbf{t}_1 \cdot 2^{pk_{\text{drop}}} + \mathbf{u}. \quad (4)$$

Step 2: Using the key-generation relation. From the key-generation algorithm, we have $\left\lfloor \frac{p}{q} \mathbf{A} \mathbf{s}_1 \right\rfloor = \mathbf{t} = \mathbf{t}_1 \cdot 2^{pk_{\text{drop}}} + \mathbf{t}_0$, where $\|\mathbf{t}_0\|_\infty \leq 2^{pk_{\text{drop}}-1}$. Substituting this value into (4) and letting $\boldsymbol{\xi} := \left\lfloor \frac{p}{q} \mathbf{A} \mathbf{z} \right\rfloor - \frac{p}{q} \mathbf{A} \mathbf{z}$ denote the rounding error (so $\|\boldsymbol{\xi}\|_\infty \leq 1$), one obtains

$$\left\lfloor \frac{p}{q} \mathbf{A} \mathbf{z} \right\rfloor - c \mathbf{t}_1 \cdot 2^{pk_{\text{drop}}} + \mathbf{u} = \frac{p}{q} \left[\mathbf{A}, \frac{q}{p} \mathbf{t} + \mathbf{v}, \mathbf{I}_k \right] \begin{bmatrix} \mathbf{z} \\ -c \\ \frac{q}{p} \mathbf{u}' \end{bmatrix}, \quad (5)$$

where we set $\mathbf{u}' := c(\mathbf{t}_0 + \frac{p}{q} \mathbf{v}) + \mathbf{u} + \boldsymbol{\xi}$.

Step 3: Bounding $\|\mathbf{u}'\|_\infty$. Since $\mathbf{v}$ is uniform over $\mathcal{R}_{q/p}^k$, we have $\|\mathbf{t}_0 + \frac{p}{q} \mathbf{v}\|_\infty \leq 2^{pk_{\text{drop}}-1} + 1$. Using $\|c\|_1 = \kappa$, $\|\mathbf{u}\|_\infty \leq 2\bar{\gamma}_2 + 1$, and $\|\boldsymbol{\xi}\|_\infty \leq 1$, we can bound the norm of $\mathbf{u}'$ as:

$$\|\mathbf{u}'\|_\infty \leq \kappa(2^{pk_{\text{drop}}-1} + 1) + 2\bar{\gamma}_2 + 1 < 4\bar{\gamma}_2 = 4\frac{p}{q}\gamma_2.$$

Step 4: Assembling the SelfTargetMSIS solution. Setting $\mathbf{t}' := \frac{q}{p} \mathbf{t} + \mathbf{v}$ and $\mathbf{H}'(\mu \parallel x) := \mathbf{H}(\mu \parallel \frac{1}{2\bar{\gamma}_2} x)$, the forgery satisfies

$$\mathbf{H}'\left(\mu \parallel \left[\mathbf{A}, \mathbf{t}', \mathbf{I}_k \right] \begin{bmatrix} \mathbf{z} \\ -c \\ \frac{q}{p} \mathbf{u}' \end{bmatrix}\right) = c, \quad (6)$$

with $\|\mathbf{z}\|_\infty < \gamma_1 - \beta_1$, $\|c\|_\infty = 1$, and $\|\mathbf{u}'\|_\infty < 4\frac{p}{q}\gamma_2$. Since $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{k \times l}$ and $\mathbf{t}' \xleftarrow{\$} \mathcal{R}_q^k$ are both uniformly distributed, $\mathbf{y} := \left[\mathbf{z}^\top \mid c \mid \frac{q}{p}\mathbf{u}'^\top \right]^\top$ constitutes a valid solution to the $\text{SelfTargetMSIS}_{\text{pp}_2}$ instance held by $\mathcal{C}$. Therefore

$$\Pr[\text{Game}_2] \leq \text{Adv}_{\text{SelfTargetMSIS}}^{\text{pp}_2}(\mathcal{C}).$$

Combining the games. Chaining the three games together gives

$$\begin{aligned} \text{Adv}_{\text{ATLAS}}^{\text{EUF-KOA}}(\mathcal{QA}) &= \Pr[\text{Game}_0] \\ &\leq \Pr[\text{Game}_1] + \text{Adv}_{\text{MLWR}}^{\text{pp}_1}(\mathcal{B}) && \text{(by the MLWR transition)} \\ &\leq \text{Adv}_{\text{SelfTargetMSIS}}^{\text{pp}_2}(\mathcal{C}) + \text{Adv}_{\text{MLWR}}^{\text{pp}_1}(\mathcal{B}), && \text{(by the SelfTargetMSIS reduction)} \end{aligned}$$

which is exactly the bound stated in the lemma.

D Analysis of Failure Probability: Rejection Sampling

Notations. For the rejection analysis, $\Pr$, $\mathbb{E}$, and Var respectively denote probability, expectation, and variance; $\mathcal{U}$, $\mathcal{N}(0, v)$, and $\mathcal{B}(N, p_h)$ denote uniform, zero-mean Gaussian (variance v), and binomial distributions. We write F_X for the CDF of X and f_X for its density when X is continuous; a distribution may replace X in the subscript. Under the continuous-uniform approximation for rounding errors, $\kappa' = \kappa/12$ is the modelled variance of a coefficient of cs_2 . Finally, $p_h = \Pr[h = 1]$ is the probability of a non-zero hint coefficient, and Event_{RS} denotes acceptance in one signing attempt. Here, we present a detailed analysis of rejection sampling.

We recall the events mentioned in Section 3.3. The rejection conditions, as defined in Figure 3, are summarised as follows:

- **Event₁ (Line 19):** $\|\mathbf{z}\|_\infty < \gamma_1 - \beta_1$.
- **Event₂ (Line 19):** $\|\mathbf{r}_0\|_\infty < \bar{\gamma}_2 - \beta_2$.
- **Event₃ (Line 19):** $\|[\text{cs}_2] - \boldsymbol{\nu}\|_\infty \leq \beta_2$ (ensuring $\mathbf{r}_1 = \mathbf{w}_1$).
- **Event₄ (Line 22):** $\|\text{ct}_0\|_\infty < \bar{\gamma}_2$.
- **Event₅ (Line 22):** $\text{HW}(\mathbf{h}) < \omega$.

Analysing Event₁. We begin by estimating the probability of **Event₁**, which is defined as.

$$\Pr[\text{Event}_1] = \Pr[\|\mathbf{z}\|_\infty < \gamma_1 - \beta_1]. \tag{7}$$

To evaluate $\Pr[\text{Event}_1]$, we analyze the distribution of the coefficients of $\mathbf{z}$. Recall from Line 19 of the signing algorithm that $\mathbf{z} = \mathbf{y} + \text{cs}_1$, where $\mathbf{y} \in \mathcal{S}_{\gamma_1-1}^l$. Thus, each coefficient of the element $(\mathbf{y})_i$ is uniformly distributed over an interval of size $2\gamma_1 - 1$, where $(\mathbf{y})_i$ denotes the i -th element of $\mathbf{y}$. Let $\bar{k}$ denote a coefficient of an element of cs_1 . The condition $\|\mathbf{z}\|_\infty < \gamma_1 - \beta_1$ implies that every coefficient of $\mathbf{z}$ lies in the interval $(-\gamma_1 + \beta_1 + 1, \gamma_1 - \beta_1 - 1]$. Using $\mathbf{z} = \mathbf{y} + \text{cs}_1$, this is equivalent to requiring that the corresponding coefficient of $(\mathbf{y})_i$ belongs to $(-\gamma_1 + \beta_1 + 1 - \bar{k}, \gamma_1 - \beta_1 - 1 - \bar{k}]$, which contains exactly $2(\gamma_1 - \beta_1) - 1$ admissible values. Therefore, the probability that a coefficient satisfies the condition is $\frac{2(\gamma_1 - \beta_1) - 1}{2\gamma_1 - 1}$. Assuming independence across the nl coefficients, we obtain the following expression. For sufficiently large γ_1 , the value can be approximated as $\Pr[\text{Event}_1] = \exp(-nl\beta_1/\gamma_1)$.

$$\begin{aligned} \Pr[\text{Event}_1] &= \Pr[\|\mathbf{z}\|_\infty < \gamma_1 - \beta_1] \\ &= \left(\frac{2(\gamma_1 - \beta_1) - 1}{2\gamma_1 - 1} \right)^{nl} \\ &= \left(1 - \frac{\beta_1}{\gamma_1 - 1/2} \right)^{nl} \approx \exp(-nl\beta_1/\gamma_1). \end{aligned}$$

Analysing Event₂. We next estimate the probability of Event₂, which is defined as

$$\Pr[\text{Event}_2] = \Pr[\|\mathbf{r}_0\|_\infty < \bar{\gamma}_2 - \beta_2]. \quad (8)$$

Following the same analysis as for Event₁, we obtain the following approximation for sufficiently large $\bar{\gamma}_2$ and $\beta_2 \gg 1/2$.

$$\Pr[\text{Event}_2] \approx \exp(-nk\beta_2/\bar{\gamma}_2)$$

Analysing Event₃. We analyze the condition $\mathbf{r}_1 = \mathbf{w}_1$. Recall that Line 11 of Figure 3 computes $(\mathbf{w}_1, _) := \text{decompose_sign}(\mathbf{w}, 2\bar{\gamma}_2)$, and Line 16 computes $(\mathbf{r}_1, _) := \text{decompose_sign}(\mathbf{w} - \lfloor \mathbf{cs}_2 \rfloor + \boldsymbol{\nu}, 2\bar{\gamma}_2)$. The equality $\mathbf{r}_1 = \mathbf{w}_1$ holds whenever the term $\lfloor \mathbf{cs}_2 \rfloor - \boldsymbol{\nu}$ does not alter the high-order component. By the correctness of `decompose_sign`, this is guaranteed if $\|\lfloor \mathbf{cs}_2 \rfloor - \boldsymbol{\nu}\|_\infty \leq \beta_2$ and $\|\mathbf{r}_0\|_\infty < \bar{\gamma}_2 - \beta_2$. The latter condition has already been analyzed through Event₂. Therefore, it remains to estimate the probability of Event₃ := $\{\|\lfloor \mathbf{cs}_2 \rfloor - \boldsymbol{\nu}\|_\infty \leq \beta_2\}$, given by

$$\Pr[\text{Event}_3] = \Pr[\|\lfloor \mathbf{cs}_2 \rfloor - \boldsymbol{\nu}\|_\infty \leq \beta_2]. \quad (9)$$

By definition, $\|\boldsymbol{\nu}\|_\infty \leq 1$, we can simplify as $\|\lfloor \mathbf{cs}_2 \rfloor\|_\infty \leq \beta_2 - 1$. Defining $\bar{\beta}_2 := \beta_2 - 1$, we rewrite the event as Event₃ = $\{\|\lfloor \mathbf{cs}_2 \rfloor\|_\infty \leq \bar{\beta}_2\}$. To estimate $\Pr[\text{Event}_3]$, we analyze the distribution of the coefficients of $\mathbf{cs}_2$. Let V denote a coefficient of $\mathbf{cs}_2$, and let U_i denote a coefficient of the i -th component of $\mathbf{s}_2$, where $\mathbf{s}_2 \in \mathbf{S}_{1/2}^k$. We model the random variables $U_1, U_2, \dots, U_k$ as independent and identically distributed (*i.i.d.*), with $U_i \sim \mathcal{U}(-\frac{1}{2}, \frac{1}{2})$. Consequently, $\text{Var}(U_i) = \frac{(\frac{1}{2} - (-\frac{1}{2}))^2}{12} = \frac{1}{12}$. Since each coefficient of c contains exactly κ non-zero entries, a coefficient of $\mathbf{cs}_2$ can be viewed as the sum of κ independent random variables distributed as U_i . Therefore, by the Central Limit Theorem, for sufficiently large κ , the random variable V can be approximated by $V \sim \mathcal{N}(0, \kappa')$, where $\kappa' = \kappa \cdot \text{Var}(U_i) = \frac{\kappa}{12}$. Using this approximation, the probability of Event₃ can be estimated as

$$\Pr[\text{Event}_3] = \Pr[\|\lfloor \mathbf{cs}_2 \rfloor\|_\infty < \bar{\beta}_2] \approx (1 - 2F_{\mathcal{N}(0, \kappa')}(-\bar{\beta}_2))^{nk}. \quad (10)$$

Here $F_{\mathcal{N}(0, \kappa')}$ denotes the cumulative distribution function (CDF) of the Gaussian distribution $\mathcal{N}(0, \kappa')$.

For ATLAS, the parameter β_2 is selected so that the event $\|\lfloor \mathbf{cs}_2 \rfloor\|_\infty < \bar{\beta}_2$ occurs with overwhelming probability. In particular, the target acceptance probability is chosen to be at least $(1 - 2^{-10})$.

Analysing Event₄. We calculate the probability of the event Event₄. The public-key component $\mathbf{t}$ is decomposed as $\mathbf{t} = \mathbf{t}_1 \cdot 2^{pk_{drop}} + \mathbf{t}_0$, and $\|\mathbf{t}_0\|_\infty \leq 2^{pk_{drop}} - 1$. Here each coefficient U_i of $\mathbf{t}_0$ is modeled as i.i.d. $\mathcal{U}(-2^{pk_{drop}-1}, 2^{pk_{drop}-1})$ with $\text{Var}(U_i) = \frac{2^{2pk_{drop}}}{12}$.

Each coefficient V of $\mathbf{ct}_0$ is the sum of κ independent variables distributed as U_i . By the Central Limit Theorem, for sufficiently large κ , V is approximated by a Gaussian distribution $V \sim \mathcal{N}(0, \text{Var}(V))$, where $\text{Var}(V) = \kappa \cdot \text{Var}(U_i) = \frac{\kappa \cdot 2^{2pk_{drop}}}{12}$. The probability that a single coefficient satisfies $|V| < \bar{\gamma}_2$ is $1 - 2F_{\mathcal{N}(0, \text{Var}(V))}(-\bar{\gamma}_2)$, where $F_{\mathcal{N}(0, \text{Var}(V))}$ denotes the CDF of $\mathcal{N}(0, \text{Var}(V))$. Assuming independence across all nk coefficients, we obtain:

$$\Pr[\text{Event}_4] \approx (1 - 2F_{\mathcal{N}(0, \text{Var}(V))}(-\bar{\gamma}_2))^{nk}. \quad (11)$$

For ATLAS, parameters are selected such that $\kappa \cdot 2^{pk_{drop}-1} < 2\bar{\gamma}_2$. Given the bound $\|\mathbf{ct}_0\|_\infty \leq \|\mathbf{c}\|_1 \cdot \|\mathbf{t}_0\|_\infty$, this selection ensures that the tail probability $F_{\mathcal{N}(0, \text{Var}(V))}(-\bar{\gamma}_2)$ remains negligible. Consequently, $\Pr[\text{Event}_4]$ is negligibly close to one.

Analysing Event₅. To estimate this probability $\Pr[\text{Event}_5]$, we first analyze the distribution of a single coefficient of $\mathbf{h}$. Let U_1 and U_2 denote coefficients of $\mathbf{r}_0$ and $\mathbf{ct}_0$, respectively, and define $V := U_1 + U_2$. By construction, the coefficient h is determined by the output of the decomposition procedure. More precisely, a hint bit is generated whenever the perturbation caused by V changes the high-order component produced by `decompose_sign`. Equivalently,

$$h = \begin{cases} 1, & \text{if } |V| > \bar{\gamma}_2, \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, the first step is to estimate the probability $\Pr[h = 1] = \Pr[|V| > \bar{\gamma}_2]$. From the analysis of Event₂ and Event₄, we have already established that $\|\mathbf{r}_0\|_\infty < \bar{\gamma}_2 - \beta_2$, and $\|\mathbf{ct}_0\|_\infty < \bar{\gamma}_2$. Consequently, we assume $U_1 \sim \mathcal{U}(-(\bar{\gamma}_2 - \beta_2), \bar{\gamma}_2 - \beta_2)$.

Let f_{U_1} and f_{U_2} denote the probability density functions of U_1 and U_2 , respectively. Since $V = U_1 + U_2$, the density of V is as follows:

$$\begin{aligned} f_V(v) &= \int_{v-(\bar{\gamma}_2-\beta_2)}^{v+(\bar{\gamma}_2-\beta_2)} f_{U_1}(v-u_2) f_{U_2}(u_2) du_2 \\ &= \frac{1}{2(\bar{\gamma}_2-\beta_2)} \int_{v-(\bar{\gamma}_2-\beta_2)}^{v+(\bar{\gamma}_2-\beta_2)} f_{U_2}(u_2) du_2 \\ &= \frac{1}{2(\bar{\gamma}_2-\beta_2)} \left(F_{U_2}(v+(\bar{\gamma}_2-\beta_2)) - F_{U_2}(v-(\bar{\gamma}_2-\beta_2)) \right), \end{aligned}$$

where F_{U_2} denotes the cumulative distribution function of U_2 .

Using the above density, the probability $\Pr[h = 1]$ can be estimated by evaluating the tail probability $\Pr[|V| > \bar{\gamma}_2]$. Subsequently, assuming independence across coefficients, the distribution of $\text{HW}(\mathbf{h})$ can be approximated by a binomial distribution, which yields an estimate for $\Pr[\text{Event}_5]$.

We next derive the CDF of the random variable V . Integrating the density function obtained above yields

$$F_V(v) = \int_{-\infty}^v f_V(u) du = \frac{1}{2(\bar{\gamma}_2-\beta_2)} \int_{v-(\bar{\gamma}_2-\beta_2)}^{v+(\bar{\gamma}_2-\beta_2)} F_{U_2}(u) du. \quad (12)$$

Here, f_{U_1} , f_{U_2} , and f_V denote the probability density functions of U_1 , U_2 , and V , respectively, while F_{U_1} , F_{U_2} , and F_V denote their corresponding cumulative distribution functions. Using the symmetry of V , the probability that a coefficient of the hint vector is non-zero can be expressed as

$$p_h = \Pr[|V| > \bar{\gamma}_2] = 2F_V(-\bar{\gamma}_2) = \frac{1}{\bar{\gamma}_2-\beta_2} \int_{-2(\bar{\gamma}_2-\beta_2)}^0 F_{U_2}(u) du. \quad (13)$$

From the analysis of Event_4 , the random variable U_2 is approximated by a Gaussian distribution with variance $\text{Var}(U_2) = 2^{2p_{k_{drop}}} \cdot \kappa'$, where $\kappa' = \kappa/12$. Consequently,

$$p_h = \frac{1}{\bar{\gamma}_2-\beta_2} \int_{-2(\bar{\gamma}_2-\beta_2)}^0 F_{\mathcal{N}(0, 2^{2p_{k_{drop}}} \kappa')}(u) du. \quad (14)$$

Having estimated p_h , we now analyze the distribution of the Hamming weight of the hint vector $\mathbf{h}$. Since each coefficient contributes independently to the hint generation process, the random variable $\text{HW}(\mathbf{h})$ can be approximated by a binomial distribution $\text{HW}(\mathbf{h}) \sim \mathcal{B}(nk, p_h)$. Let $F_{\mathcal{B}(nk, p_h)}$ denote the corresponding cumulative distribution function. Then the probability of Event_5 can be approximated by

$$\Pr[\text{Event}_5] = \Pr[\text{HW}(\mathbf{h}) < \omega] \approx F_{\mathcal{B}(nk, p_h)}(\omega). \quad (15)$$

For ATLAS, the parameter ω is selected so that the event $\text{HW}(\mathbf{h}) < \omega$ holds with overwhelming probability. In particular, the target acceptance probability is chosen to be at least $1 - 2^{-10}$.

Analysing the whole event Event_{RS} . Combining the above analysis, we saw that $\Pr[\text{Event}_3]$, $\Pr[\text{Event}_4]$, and $\Pr[\text{Event}_5]$ are all negligibly close to one. Consequently, the overall acceptance probability is dominated by Event_1 and Event_2 . Therefore,

$$\Pr[\text{Event}_{\text{RS}}] \approx \Pr[\text{Event}_1] \cdot \Pr[\text{Event}_2] \approx \exp\left(-n \left(\frac{\beta_1 l}{\gamma_1} + \frac{\beta_2 k}{\bar{\gamma}_2} \right)\right). \quad (16)$$

Hence, the expected number of iterations of the rejection-sampling loop is approximately

$$\frac{1}{\Pr[\text{Event}_{\text{RS}}]} = \exp\left(n \left(\frac{\beta_1 l}{\gamma_1} + \frac{\beta_2 k}{\bar{\gamma}_2} \right)\right). \quad (17)$$

E Detailed Algebraic Security Analysis of ATLAS

Here, we provide a complete algebraic security analysis, as sketched in Section 3.4.

Following the design methodology of Dilithium, we instantiate and evaluate our concrete hardness with necessary adaptations for the Module Learning With Rounding (MLWR) [7,30] and Module Short Integer Solution (MSIS) problems. Because no known cryptanalytic techniques exploit the underlying algebraic module structure more effectively than unstructured lattice reductions, we map our MLWR and MSIS instances to standard LWR and SIS instances over the scalar ring $\mathbb{Z}_q$.

Security of MLWR. An instance of $\text{MLWR}_{q,p,l,k}$ is represented as an unstructured LWR instance of dimension $nk \times nl$. Concretely, solving $\text{MLWR}_{q,p,l,k}$ corresponds to recovering $\text{vec}(\mathbf{s}) \in \mathbb{Z}^{nl}$ from the pair $(\text{rot}(\mathbf{A}), \text{vec}(\mathbf{t})) \in \mathbb{Z}_q^{nk \times nl} \times \mathbb{Z}_p^{nk}$, where $\text{vec}(\cdot)$ maps a vector of polynomials in $\mathcal{R}_q$ to the concatenated vector of their coefficient embeddings. The matrix $\text{rot}(\mathbf{A}) \in \mathbb{Z}_q^{nk \times nl}$ is formed by replacing each polynomial entry $a_{ij} \in \mathcal{R}_q$ with its $n \times n$ anti-circulant rotation matrix whose z -th column is $\text{vec}(x^{z-1} \cdot a_{ij})$.

Given an LWR sample $(\mathbf{A}, \mathbf{t} := \lfloor \frac{p}{q} \mathbf{A} \mathbf{s} \rfloor)$, we embed it into an equivalent LWE instance by rescaling:

$$\frac{q}{p} \mathbf{t} = \mathbf{A} \mathbf{s} + \frac{q}{p} \boldsymbol{\xi},$$

where $\boldsymbol{\xi} := \lfloor \frac{p}{q} \mathbf{A} \mathbf{s} \rfloor - \frac{p}{q} \mathbf{A} \mathbf{s}$ denotes the deterministic rounding error vector. Under the standard heuristic assumption, each coordinate of $\boldsymbol{\xi}$ is distributed independently and uniformly over $(-\frac{1}{2}, \frac{1}{2})$, yielding an induced LWE noise variance of

$$\text{Var} = \left(\frac{q}{p}\right)^2 \cdot \frac{1}{12} = \frac{q^2}{12p^2}.$$

We estimate the concrete hardness expressed as the minimal BKZ block size b required for recovery using standard lattice estimators parameterized by dimension nl , modulus q , and error variance Var .

Security of SelfTargetMSIS. As established in the unforgeability argument, a successful signature forgery directly yields a non-trivial solution to the Self-Target MSIS (SelfTargetMSIS) relation. Specifically, breaking the signature scheme requires finding an input $\mathbf{w}$ such that the random oracle evaluates to $H'(\mu \parallel \mathbf{w}) = c$, followed by constructing a valid signature pair $(\mathbf{z}, \mathbf{u}')$ that satisfies the linear relation

$$\mathbf{w} = \mathbf{A} \mathbf{z} - c \frac{q}{p} \mathbf{t} + \frac{q}{p} \mathbf{u}'.$$

Rearranging this identity yields

$$\mathbf{A} \mathbf{z} + \frac{q}{p} \mathbf{u}' = \mathbf{t}', \quad \text{where } \mathbf{t}' := \mathbf{w} + c \frac{q}{p} \mathbf{t}.$$

Hence, forging a signature reduces to identifying short vectors $\mathbf{z}$ and $\mathbf{u}'$ satisfying the above inhomogeneous relation while respecting the canonical verification bounds:

$$\|\mathbf{z}\|_\infty < \gamma_1 - \beta_1 \quad \text{and} \quad \|\mathbf{u}'\|_\infty < 4 \frac{p}{q} \gamma_2.$$

This relation directly induces an inhomogeneous $\text{MSIS}_{k,l+1,\zeta}$ instance over the augmented matrix $[\mathbf{A} \mid \mathbf{t}']$, where the target euclidean norm bound ζ is constrained by the admissible infinity norms of $\mathbf{z}$ and $\mathbf{u}'$.

Reduction to Standard MSIS. The augmented relation $[\mathbf{A} \mid \mathbf{t}']$ defines an instance of $\text{MSIS}_{k,l+1,\zeta}$. To assess its resistance against lattice reduction, we map this module problem to an equivalent scalar SIS instance over $\mathbb{Z}_q$. Expanding $[\mathbf{A} \mid \mathbf{t}']$ via its canonical coefficient representation yields the rotation matrix $\text{rot}([\mathbf{A} \mid \mathbf{t}']) \in \mathbb{Z}_q^{nk \times n(l+1)}$, defining an SIS problem of dimension $nk \times n(l+1)$. Similarly, solving the homogeneous relation on the public matrix alone corresponds to $\text{rot}(\mathbf{A}) \in \mathbb{Z}_q^{nk \times nl}$.

Because our choice of modulus q , dimensions (k, l) , and acceptance bounds (γ_1, γ_2) closely follow the design constraints of Dilithium, the resulting MSIS instances exhibit strictly comparable hardness profiles, with minor deviations arising solely from our modulus ratio q/p and dimension adjustments.

BKZ Reduction Mechanics. The most effective strategy for solving the induced LWE and SIS instances is lattice basis reduction via the Block, Korkine, Zolotarev (BKZ) algorithm [37]. BKZ systematically queries a Shortest Vector Problem (SVP) oracle inside a projected subspace of block size b . Because the operational cost of solving SVP grows exponentially with b , finding an admissible vector becomes computationally intractable for sufficiently large block sizes. Table 10 summarizes the asymptotic time complexities of state-of-the-art classical, quantum, and conservative sieve algorithms in dimension b .

Table 10: Asymptotic cost of state-of-the-art SVP solvers in dimension b .

Setting	Asymptotic Complexity
Classical [9]	$\approx 2^{0.292 b}$
Quantum [29]	$\approx 2^{0.265 b}$
Paranoid [21]	$\approx 2^{0.2075 b}$

The paranoid estimate serves as a conjectured geometric lower bound rather than an instantiated algorithm; it corresponds to packing the b -dimensional unit sphere with spherical cones of angular radius $\pi/3$, which demands $\approx 2^{0.2075 b}$ cones. Overcoming this bound would necessitate fundamental breakthroughs in sphere packing and the geometry of numbers [21].

Primal and Dual Attack Formulations. We evaluate concrete gate complexities against both primal and dual BKZ reductions:

- **Primal Attack (uSVP Embedding):** Given an LWE instance $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{\bar{m} \times \bar{n}} \times \mathbb{Z}_q^{\bar{n}}$ satisfying $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$ with target vector norm $\|\mathbf{u}\|_2 = \|(\mathbf{e}, \mathbf{s}, 1)\|_2 = \epsilon$, the attack embeds the instance into a lattice of dimension $\bar{m} + \bar{n} + 1$. Solving $\text{LWE}_{\bar{m}, \bar{n}, \epsilon}$ requires finding a minimal block size $b \geq 50$ such that $\epsilon\sqrt{b} \leq \delta^{2b - (\bar{m} + \bar{n} + 1)} \cdot q^{\bar{n}/(\bar{m} + \bar{n} + 1)}$, where $\delta = \left(\frac{(\pi b)^{1/b} \cdot b}{2\pi e}\right)^{\frac{1}{2(b-1)}}$ represents the root Hermite factor of BKZ- b . The optimal number of samples $\bar{m}$ is chosen to minimize the overall sieving cost $2^{0.292b}$.
- **Dual Attack (Distinguisher):** The dual attack searches for a short non-zero vector $\mathbf{v} \in \mathbb{Z}_q^{\bar{m}}$ in the dual lattice such that $\mathbf{v}^\top \mathbf{A} \equiv \mathbf{0} \pmod{q}$. Computing the inner product $\mathbf{v}^\top \mathbf{b} = \mathbf{v}^\top \mathbf{e} \pmod{q}$ yields a statistical distinguishing advantage when $\|\mathbf{v}\|_2$ is sufficiently small. The required block size b is the smallest integer ≥ 50 satisfying $-2\pi^2 \tau(b)^2 \geq \ln(2^{-0.2075b/2})$, where $\tau(b) := \delta^{\bar{m} + \bar{n} - 1} \cdot q^{\bar{n}/(\bar{m} + \bar{n})} \cdot \frac{\epsilon}{q}$.